Ant Colony Optimization and Local Improvement

H. Crauwels
De Nayer instituut
Hogeschool voor Wetenschap & Kunst
J. De Nayerlaan 5
B-2860 Sint-Katelijne-Waver, Belgium
hcr@denayer.wenk.be

D. Van Oudheusden
Centre for Industrial Management
K.U.Leuven
Celestijnenlaan 300A
B-3001 Heverlee, Belgium
Dirk.VanOudheusden@cib.kuleuven.ac.be

November, 2003

Abstract

Ant colony optimization is a metaheuristic that has proven its quality and versatility on various combinatorial optimization problems. One reason for this good performance is the integration of a local search procedure. This paper investigates the relative contribution of the basic ant system and the local improvement techniques. For the computational experiments, instances of the traveling tournament problem are used. It is a sports timetabling problem proposed to abstract the salient features of major league baseball in the United States. This is a double round robin tournament for n teams with no repeaters. Because of the large distances between the home cities, a limited number of consecutive home or away games is allowed. The objective is to construct a timetable with minimum travel distance. Even small instances of this problem seem to be very difficult to solve.

Keywords : Traveling Tournament Problem, Sports Timetabling, Local Search, Ant Colony Optimization, Neighbourhood Search.

1 Introduction

Ant algorithms are inspired by the observation of real ant colonies. Real ants searching for food are capable to find the shortest path between their nest and a food source by exchanging information via pheromones. This substance is deposited by ants on the ground while walking from the nest to a food source and vice versa. In this way a pheromone trail is formed. As other ants observe the pheromone trail and are attracted to it, the path is marked again and will attract even more ants to follow the trail.

In ant colony optimization algorithms (ACO), a finite sized colony of artificial ants collectively searches for good quality solutions to the optimization problem under consideration. Each ant builds a solution in a finite number of steps whereby in each step the partial solution is extended with one additional element. The selection of this additional element is influenced through problem-specific heuristic information, called visibility, as well as pheromone trails, an indication of how good the choice of that element was in former runs. After an artificial ant has constructed a feasible solution, the pheromone trails are updated depending on the objective function value of the constructed solution. This update will influence the selection process of the next ants. An overview of ant algorithms can be found in Dorigo *et al.* [2].

For several classical combinatorial optimization problems, good results are reported with an ant colony optimization approach. In several of these ACO algorithms, a local search procedure is integrated. Such a local search procedure is applied to the solutions generated by the artificial ants before updating the pheromone trails; and this local search is largely responsible for the good performance of the algorithm. However, a comparison of results obtained without and with local improvement techniques is mostly not reported. This paper investigates the relative contribution of the basic ant system and some local improvement techniques.

For the computational experiments, instances of the traveling tournament problem (TTP) are used. The reason for this choice is twofold. Firstly, the TTP is a rather hard problem to solve, mainly because of the mix of feasibility and optimality issues. Therefore, considering the application of a metaheuristic is worthwhile. Secondly, the TTP is an example of a constraint satisfaction problem (CSP), a rather new area for ACO algorithms.

2 The traveling tournament problem

In a sport competition, n given teams play against each other over a period of time according to a certain scheme. An example of such a scheme is double round robin. It determines that every team l plays twice against every other team: a *home* match when team l uses its own facilities and an *away* match when the match takes place at the facility of the opponent. When the number of teams, n, is even, and every team plays at most one match per date, the minimal number of dates over which the $n(n-1)$ matches are distributed, is equal to $M = 2n - 2$. In a classical tournament, one tries to minimize the number of *breaks*, i.e. the fact that a team plays two consecutive matches in the same location, where the term location denotes either home or away. Because of the attractiveness of the tournament, the situation that two teams play their two matches against each other on two consecutive dates should be avoided. Such a situation is called a *repeater*.

In this paper, we consider the traveling tournament problem (Easton *et al.* [3]). This is a double round robin tournament for n teams, with n even and with no repeaters. Breaks are allowed because in a large country (e.g. USA), where the distances between the home cities can be very large, it is advantageous to organize a *tour* for a number of consecutive away matches. However, no more than three consecutive away games (*road trips*) and also no more than three consecutive home games (*home stands*) are allowed for any team. The objective is to minimize the sum of the lengths of the tours traveled by the teams, where we assume that all teams start and end in their home city and that the distance matrix (d_{ij} with $1 \leq i, j \leq n$) is symmetric.

A general technique to solve such a highly constrained problem is backtracking. It is a trial-and-error process that gradually builds up and scans (prunes) a tree of subtasks. At each step in the search tree, a list of candidates of teams for the next match of a specific team is constructed. The first candidate of the list is selected and the procedure continues analogously with the next match for the next team until some constraint cannot be satisfied. Then, backtracking occurs and the next element from the candidate list is selected. This process continues until a first complete timetable is built.

3 The metaheuristic

In our heuristic for the traveling tournament problem, we use a colony of n ants. These n ants perform a number of iterations. During each iteration t, each ant k constructs a timetable $T^k(t)$ by starting with the determination of the first opponent of team k and continuing with determining opponents for teams $k+1, \ldots, n, 1, \ldots, k-1$. Then the opponents of the second round are determined and this is continued until a complete timetable is constructed.

The complete timetable $T^k(t)$ is equal to the union of n sequences $T_l^k(t)$, where each sequence consists of the $2(n-1)$ opponents of team l. In such a sequence, a positive number i indicates that team l plays at home against team i; and a negative number $-i$ means that team l plays away against team i. The total distance traveled $L^k(t)$ is equal to the sum of the distances traveled by each team:

$$L^k(t) = \sum_{l=1}^{n} P_l^k \qquad \text{(with } P_l^k \text{ the length of the tour } T_l^k(t)\text{)} \tag{1}$$

For the determination of an opponent for a team, a list of candidates is constructed. The order in which this list of candidates is searched, is influenced through heuristic information (η_{ij}) as well as pheromone trails (τ_{ij}). The former is an indicator of how good the choice of that match *seems to be*, and the latter indicates how good the choice of that match *was* in former runs.

An example of a timetable for $n = 4$ teams and the corresponding graph that is "traveled" by an ant, is given in Figure 1. In this graph, only the path starting at $i = 3$ (the matches of team 3) is shown.

Team\ Round	1	2	3	4	5	6
1	2	3	4	-2	-3	-4
2	-1	-4	3	1	4	-3
3	4	-1	-2	-4	1	2
4	-3	2	-1	3	-2	1

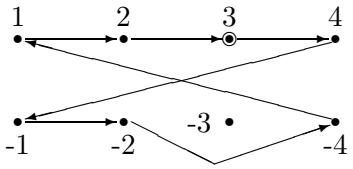

Figure 1: a timetable for $n = 4$

This underlying graph consists of $2n$ nodes $(-1, -2, \ldots, -n, 1, 2, \ldots, n)$ and each ant k constructs n paths through this graph. Each path starts at node i, visits every other node $j \neq |i|$ and ends at node i. In such a path, there are four types of arcs: (1) between two nodes each labeled with a positive number, indicating two consecutive home matches; except when one of the two nodes is equal to i: these arcs correspond to the first and last match in the sequence which are home matches; (2) between two nodes with negative numbers, two consecutive away matches; (3) between a node with a positive number and a node with a negative number, a home match followed by an away match or the first match in the sequence is an away match; (4) between a node with a negative number and a node with a positive number, an away match followed by a home match or the last match in the sequence is an away match. Because the underlying graphs consists of $2n$ nodes, we use $2n \times 2n$ matrices for the visibility (η_{ij}) and the pheromone trail (τ_{ij}).

The most obvious heuristic information to be used is the distance between two cities. This

distance however is of no importance for two consecutive home matches. Therefore,

$$\eta_{ij} = \begin{cases} 1.0 & \text{for } 1 \leq i, j \leq n \\ Q/d_{|i||j|} & \text{otherwise, with } Q \text{ a constant} \end{cases} \quad (2)$$

During the first iteration, some small constant value is used for each pheromone trail. At the end of each iteration, when each ant has constructed its timetable, the pheromone trails are updated. The pheromone trail between node i and node j on a path in the underlying graph is incremented with a value depending on the total distance traveled ($L^{(k)}(t)$).

4 Local improvement techniques

As indicated in the introduction, ACO algorithms have been hybridized with local improvement techniques. For the TTP, such a technique can be easily integrated. Instead of stopping the backtracking algorithm at the first complete feasible solution, an ant continues with the search and it can find a number of additional solutions. Of course, this backtracking cannot continue until all solutions have been found because this would take too much computation time. The procedure is stopped when E solutions have been calculated since the last best solution. When the parameter E is small, only a few additional solutions will be calculated. With a large parameter E, the last part of the tournament will be quite good.

A second intensifying device is based on neighbourhood search. A small modification is applied to the tournament that is constructed by an ant. The new tournament is first checked on feasiblity: no repeaters and no more than three consecutive home or road games are allowed for any team. For a feasible tournament the total distance traveled is calculated and when this total distance is smaller than that of the original tournament, the neighbour is accepted. Generating new neighbours is continued on this new better tournament. The process of generating new neighbours terminates when no move out of the whole neighbourhood results in a better feasible tournament. A number of different neighbourhood structures have been considered: swapping the home/away roles of two teams, swapping two rounds, inserting one round after another round and swapping two teams.

5 Computational experience

For the computational experiments, we use the instances of the Challenge Traveling Tournament web page at http://mat.gsia.cmu.edu/TOURN/. It concerns a series of instances from the Major League Baseball (USA), which provided the original inspiration to formulate this problem. Easton *et al.* [3] have generated the National League distance matrices by using "air distance" from the city centers. To generate smaller instances, they simply take subsets of the teams, resulting in instances NL4, NL6, NL8, NL10, NL12, NL14 and NL16. The number indicates the number of teams in the instance taken from a listing of teams ordered roughly from north-east to south-west. Easton *et al.* [4] can solve instances up to $n = 8$ teams, using parallel programming with 20 processors and 4 to 5 days of computation time.

Table 1 presents an overview of some partial computational results. Several heuristic approaches have been suggested for the traveling tournament problem. The simulated annealing procedure of Anagnostopoulos *et al.* [1] produces at the time of writing the best known solutions. These values are reported in the column "Best". To investigate the effectiveness of ant colony optimization (column "ACO"), a comparison is made with the results obtained with

a heuristic where the pheromone trails are replaced by random numbers (column "random"). The last two columns give results for ACO with local improvement techniques integrated. For the column "ACO+B", the basic backtracking procedure that is used by each ant, is continued until $E = 5000$ solutions have been calculated. The column "ACO+B+NS" shows the solution values when also neighbourhood search techniques are integrated.

Table 1: computational results

	Best	random	ACO	ACO+B	ACO+B+NS
NL6	23916	26130	26073	25072	23916
NL8	39721	47585	47161	43231	40797
NL10	59583	77311	74929	70787	67871
NL12	112800	142239	144373	139230	128909
NL14	190368	274918	264069	263535	240445
NL16	267194				346530

The number of iterations for each technique is 25. Problems with a limited number of cities ($n = 6, 8$) require a few minutes CPU time, whereas for the larger problems the procedure takes a few days. Despite these long computation times, the results obtained by the ACO procedure cannot compete with the best known solution values.

6 Concluding remarks

The traveling tournament problem is a very hard problem, because of the mix of feasibility and optimality. The feasibility issue concerns sufficiently varying the home and away pattern so as to avoid long home stands and road trips. The second requirement is preventing excessive travel by minimizing total travel distance. The reported computational results of ant colony optimization for this problem are not very promising. A possible explanation is that ant colony optimization may be a good method for the optimization part of the problem but that it cannot cope very well with the feasibility issue.

References

[1] Anagnostopoulos, A., L. Michel, P. Van Hentenryck, and Y. Vergados (2003), "A Simulated Annealing Approach to the Traveling Tournament Problem," *Proceedings of CP'AI'OR'03*, Montreal, Canada.

[2] Dorigo, M., G. Di Caro, and L.M. Gambardella (1999), "Ant Algorithms for Discrete Optimization," *Artificial Life*, Vol.5, No.2, pp. 137–172.

[3] Easton, K., G.L. Nemhauser, and M.A. Trick (2001), "The Traveling Tournament Problem: Description and Benchmarks," *Principles and Practice of Constraint Programming — CP 2001*, Springer Lecture Notes in Computer Science 2239, pp. 580–585.

[4] Easton, K., G.L. Nemhauser, and M.A. Trick (2003), "Solving the Traveling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach," In: Burke, E. and P. De Causmaecker (EDS.): *Practice and Theory of Automated Timetabling IV*. Springer Lecture Notes in Computer Science 2740, pp. 100–109.