The accumulated experience ant colony for the travelling salesman problem

Randall, Marcus; Montgomery, James

Published in:
International Journal of Computational Intelligence and Applications

DOI:
10.1142/S1469026803000938

The Accumulated Experience Ant Colony for the Travelling Salesman Problem*

Marcus Randall, James Montgomery**

School of Information Technology
Bond University, QLD 4229
Ph: +61 7 5595 3361
Fax: +61 7 5595 3320
{mrandall, jmontgom}@bond.edu.au

Abstract. Ant colony optimisation techniques are usually guided by pheromone and heuristic cost information when choosing the next element to add to a solution. However, while an individual element may be attractive, usually its long term consequences are neither known nor considered. For instance, a short link in a TSP may be incorporated into an ant's solution, yet, as a consequence of this link, the rest of the path may be longer than if another link was chosen. The Accumulated Experience Ant Colony uses the previous experiences of the colony to guide in the choice of elements. This is in addition to the normal pheromone and heuristic costs. The results indicate that on some problems this helps to find improved solutions to the travelling salesman problem.

Keywords: Ant colony optimisation, travelling salesman problem.

1 Introduction

Ant Colony Optimisation (ACO) meta-heuristics have proved to be remarkably successful in solving a range of discrete optimisation problems. The core principal on which these techniques operate is to use a collective memory of the characteristics of the problem being solved, built up over time. Each ant consults this memory when augmenting its solution. This memory usually stores the colony's preference for adding a particular element with respect to the current element. For instance, given a travelling salesman problem (TSP) in which a Hamiltonian circuit of minimum total length is sought, an ant would choose the next city based on the pheromone amount associated between the current city and the potential next city in conjunction with the distance. The accumulated experience ant colony (AEAC) adds another dimension to this by considering how the choice of elements affects the solution quality *after* it has been incorporated. As such, we have modified the characteristic element selection equations

* We wish to acknowledge the Australian Research Council for their financial assistance to this research.

** This author is a PhD scholar supported by an Australian Postgraduate Award.

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.
 https://doi.org/10.1142/S1469026803000938

to incorporate a weighting term for the accumulated experience component. This weighting is based on the characteristics of partial solutions generated within the current iteration. Elements that appear to lead to better solutions are valued more highly, while those that lead to poorer solutions are made less desirable.

Heusse, Gurin, Snyders and Kuntz [3] propose a similar information sharing system for routing in packet-switched networks, called *Co-operating Asymmetric Forward routing* (CAF routing). Nodes in CAF routing send out agents to calculate the current delay on network routes. The exact route each agent takes is based on the same routing information used for normal data packets, but is modified by information gathered by agents travelling on the same route in the opposite direction. Thus, CAF routing uses knowledge gathered from many agents to report on the current state of network routes. This differs from AEAC, however, which uses accumulated experience in a heuristic fashion to *estimate* the utility of individual solution elements.

This paper is organised as follows. Section 2 has a brief overview of ACO while Section 3 explains how we adapt it to incorporate the accumulated experience component. Section 4 shows the results of AEAC on some benchmark TSPs while Section 5 gives the conclusions of this work.

2 ACO

ACO is an umbrella term for a number of similar meta-heuristics [2]. The Ant Colony System (ACS) [1] meta-heuristic will be described here in order to demonstrate the underlying principals of ACO.

ACS can best be described with the TSP metaphor. Consider a set of cities, with known distances between each pair of cities. The aim of the TSP is to find the shortest path to traverse all cities exactly once and return to the starting city. ACS is applied to this problem in the following way. Consider a TSP with N cities. Cities i and j are separated by distance $d(i,j)$. Scatter m ants randomly on these cities ($m \leq N$). In discrete time steps, all ants select their next city then simultaneously move to their next city. Ants deposit a substance known as *pheromone* to communicate with the colony about the utility (goodness) of the edges. Denote the accumulated strength of pheromone on edge (i,j) by $\tau(i,j)$.

At the commencement of each time step, Equations 1 and 2 are used to select the next city s for ant k currently at city r. Equation 1 is a greedy selection technique favouring cities which possess the best combination of short distance and large pheromone levels. Equation 2 balances this by allowing a probabilistic selection of the next city.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ \tau(r,u)[d(r,u)]^{\beta} \right\} & \text{if } q \leq q_0 \\ \text{Equation 2} & \text{otherwise} \end{cases} \tag{1}$$

$$p_k(r,s) = \begin{cases} \frac{\tau(r,s)[d(r,s)]^{\beta}}{\sum_{u \in J_k(r)} \tau(r,u)[d(r,u)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.
 https://doi.org/10.1142/S1469026803000938

Note that $q \in [0,1]$ is a uniform random number and q_0 is a parameter. To maintain the restriction of unique visitation, ant k is prohibited from selecting a city which it has already visited. The cities which have not yet been visited by ant k are indexed by $J_k(r)$. It is typical that the parameter β is negative so that shorter edges are favoured. Linear dependence on $\tau(r,s)$ ensures preference is given to links that are well traversed (i.e. have a high pheromone level). The pheromone level on the selected edge is updated according to the local updating rule in Equation 3.

$$\tau(r,s) \leftarrow (1-\rho) \cdot \tau(r,s) + \rho \cdot \tau_0 \tag{3}$$

Where:

ρ is the local pheromone decay parameter, $0 < \rho < 1$.
τ_0 is the initial amount of pheromone deposited on each of the edges.

Upon conclusion of an iteration (i.e. once all ants have constructed a tour), global updating of the pheromone takes place. Edges that compose the best solution are rewarded with an increase in their pheromone level. This is expressed in Equation 4.

$$\tau(r,s) \leftarrow (1-\gamma) \cdot \tau(r,s) + \gamma \cdot \Delta\tau(r,s) \tag{4}$$

$$\Delta\tau(r,s) = \begin{cases} \frac{Q}{L} & \text{if } (r,s) \in \text{ globally best tour} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Where:

$\Delta\tau(r,s)$ is used to enforce the pheromone on the edges of the the solution (see Equation 5).
L is the length of the best (shortest) tour to date while Q is a problem dependent parameter [1].
γ is the global pheromone decay parameter, $0 < \gamma < 1$.

3 Accumulated Experience Ant Colony

The AEAC described herein is based on ACS, although it is possible to apply it in a range of ACO meta-heuristics. Essentially a new term w is incorporated into Equations 1 and 2 giving Equations 6 and 7 respectively.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ w(r,u)\tau(r,u)[d(r,u)]^{\beta} \right\} & \text{if } q \leq q_0 \\ \text{Equation 2} & \text{otherwise} \end{cases} \tag{6}$$

$$p_k(r,s) = \begin{cases} \frac{w(r,s)\tau(r,s)[d(r,s)]^{\beta}}{\sum_{u \in J_k(r)} w(r,u)\tau(r,u)[d(r,u)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Given that $u \in J_k(r)$, $w(r,u)$ represents the relative weighting of link (r,u) compared to all other choices from node r, $0 \leq w(r,u) \leq 2$. This is calculated for

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.
 https://doi.org/10.1142/S1469026803000938

all ants that have incorporated link (r, u) into their solution within the current iteration. If link (r, u) has been found to lead to longer paths after it has been incorporated into the solution, then the weighting $w(r, u) < 1$. On the other hand, if the reverse is the case, then $w(r, u) > 1$. If the colony as a whole has never incorporated link (r, u), its weighting is simply 1. Therefore, links that have historically been shown to induce shorter overall paths will be favoured over those that have not. This represents a longer term approach than just using pheromone alone.

The value of w is updated at the beginning of each step for all links that are reachable by an ant during that step. This reduces computation time without affecting the algorithm's overall behaviour, as it is not necessary to evaluate the probability (Equation 7) of links that cannot be considered by an ant. Before calculating w, the costs of the partial tours so far constructed are linearly scaled in the range [-1, 1], where -1 is the scaled value of the minimum length path and 1 is the corresponding value for the maximum length path. Then, for a given link (r, u), $w(r, u)$ is calculated by the following. The mean scaled cost of all paths that have used (r, u) is computed and, in the case of a minimisation problem such as the TSP, subtracted from 1. Hence, if (r, u) has been used predominantly in shorter paths, the mean of the scaled costs will be closer to -1 and as such the value of $w(r, u)$ will be closer to 2. In the opposite situation, where a link has been used predominantly in longer paths, the correspondingly higher mean scaled cost will yield a value of w closer to 0. It is worth noting that as the number of ants that have used (r, u) approaches m, $w(r, u)$ will in general approach 1 given a uniform distribution of partial tour costs. This is because it becomes difficult to determine the individual contribution of (r, u) to the cost of solutions. The algorithm for updating w for all links is summarised in Figure 1 while Equation 8 is for a single link. The computational overhead associated with updating w is $O(n^2)$.

$$w(r, s) = \begin{cases} 1 - \frac{1}{|A(r,s)|} \sum_{a \in A(r,s)} \frac{c(a) - c_{mid}}{c_{max} - c_{mid}} & \text{if } |A(r, s)| > 0 \\ 1 & \text{otherwise} \end{cases} \quad (8)$$

Where:

c_{min} and c_{max} are the costs of the minimum and maximum partial tours respectively.
c_{mid} is the linear mid-point between c_{min} and c_{max}.
$c(a)$ is the cost of the partial tour constructed by ant a.
$A(r, s)$ is the set of all ants that have incorporated link (r, s) into their respective solutions.

4 Computational Experience

A control strategy (normal ACS) and the AEAC are run in order to evaluate the performance of the system. Table 1 describes the TSP instances [5] with which the AEAC meta-heuristic is tested.

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.
 https://doi.org/10.1142/S1469026803000938

```
Determine c_min and c_max
c_mid ← (c_max + c_min)/2
For each ant a
    c'(a) ← (c(a) − c_mid)/(c_max − c_mid)
End For
For each link (r, s) adjacent to an ant
    If |A(r, s)| > 0 Then
        c̄'_{a∈A(r,s)} ← mean of c'(a) for all a in A(r, s)
        w(r, s) ← 1 − c̄'_{a∈A(r,s)}
    Else
        w(r, s) ← 1
    End If
End For
```

Where:

$c'(a)$ is the scaled cost of the partial solution constructed by ant a.

Fig. 1. Algorithm for calculating w for each link.

The computing platform used to perform the experiments is a 550 MHz Linux machine. The computer programs are written in the C language. Each problem instance is run across 10 random seeds. The ACS parameter settings are given in Table 2.

4.1 Results

The results are given in Tables 3 and 4. The first contains the results of the control strategy while the latter contains the AEAC results. The minimum (**Min**), median (**Med**), maximum (**Max**) and inter-quartile range (**IQR**) are used to summarise the results.

AEAC performed well on a number of the problems, achieving lower minimum costs on `bays29`, `eil51`, `st70`, `pr76` and `ch130`, and equivalent minimum costs on `gr24`, `hk48` and `eil76` compared to the control strategy. On `gr24`, `bays29` and `hk48` it found the optimal solution, while the control strategy only found the optimal for `gr24` and `hk48`. On all problems up to 200 cities, the median cost achieved by AEAC was within 3.3% of the optimal cost.

The upper bounds on costs reported by AEAC were also better than the control strategy on a number of the problems. This result, in conjunction with the generally smaller IQR exhibited by AEAC, suggests that AEAC focuses the search onto a smaller area of the solution space.

Above 100 cities, AEAC performed worse than the control strategy on all problems except `ch130`. On the problems with more than 130 cities, the median costs achieved by AEAC were between 0.5% and 8.5% of those achieved by the control strategy. However, as the number of cities increases, so does the gap

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.

Table 1. TSP instances used in this study.

Instance	Description	Optimal Cost
gr24	24 cities	1272
bays29	29 cities	2020
hk48	48 cities	11461
eil51	51 cities	426
berlin52	52 cities	7542
st70	70 cities	675
eil76	76 cities	538
pr76	76 cities	108159
kroA100	100 cities	21282
ch130	130 cities	6110
d198	198 cities	15780
kroA200	200 cities	29368
lin318	318 cities	42029
rd400	400 cities	15281
pcb442	442 cities	50778
att532	532 cities	27686

Table 2. Parameter settings used in this study.

Parameter	Value
β	-2
γ	0.1
ρ	0.1
m	10
q_0	0.9
iterations	3000

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.

Table 3. Results for control strategy.

Problem	Cost				CPU Time (seconds)			
Instance	Min	Med	Max	IQR	Min	Med	Max	IQR
gr24	1272	1278	1328	1	18	18	18	0
bays29	2026	2035	2066	7	26	26	27	0
hk48	11461	11491	11847	78	71	71	71	0
eil51	430	435	441	4	79	80	80	0
berlin52	7542	7855	8002	313	83	83	83	0
st70	686	708	739	26	150	150	150	0
eil76	546	558	564	5	198	198	198	0
pr76	108308	109878	112456	837	177	177	178	0
kroA100	21292	21393	22030	474	304	304	305	0
ch130	6265	6321	6538	129	519	520	520	0
d198	15954	16101	16253	209	1186	1186	1194	1
kroA200	29749	30318	30896	541	1230	1231	1232	1
lin318	45716	47121	48144	1162	3062	3064	3066	1
rd400	16396	16709	17493	355	4945	4946	4948	2
pcb442	60970	63025	64858	1614	5971	6024	6113	32
att532	33941	35335	36441	1498	8743	8758	8768	4

Table 4. Results for AEAC.

Problem	Cost				CPU Time (seconds)			
Instance	Min	Med	Max	IQR	Min	Med	Max	IQR
gr24	1272	1278	1279	0	101	101	102	0
bays29	2020	2030	2038	0	111	111	111	0
hk48	11461	11600	11824	0	164	164	170	0
eil51	427	432	437	0	174	175	175	0
berlin52	7547	7790	8148	0	178	178	179	0
st70	678	687	712	0	258	258	259	0
eil76	546	555	559	0	289	290	290	0
pr76	108274	109735	112725	1621	289	289	290	0
kroA100	21373	21512	21915	97	442	442	443	0
ch130	6180	6269	6407	119	692	693	693	1
d198	16044	16209	16465	175	1492	1493	1494	1
kroA200	29769	30203	31135	594	1538	1539	1544	2
lin318	46713	50651	53244	2090	3768	3771	3775	4
rd400	17922	18124	20073	189	5980	5984	5990	4
pcb442	64866	66333	68394	1640	7268	7272	7283	4
att532	36271	36989	38049	670	10512	10521	10547	8

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.

between the solutions' costs produced by AEAC and the corresponding optimal costs.

The computational overhead associated with calculating w adversely affects the CPU time used by AEAC. On all problems AEAC had longer CPU times than the control strategy. This effect is particularly evident on the smaller problems, but decreases in proportion to the inverse square of the number of cities. This decay is to be expected, however, as the procedure for determining w only considers the reachable links at each step. Since the number of ants remains constant, the number of reachable links at each step also decays with the inverse square of the number of cities.

5 Conclusions

The AEAC approach relies on historical information and patterns developed over time in order to calculate which element to add to a solution at a particular step of the algorithm. An alternative approach is to use speculative information about an element by looking ahead at all (or some) of the possible future choices.

Our AEAC performed most successfully on the smaller problems (those with less than 100 cities), finding the optimal solution to three and equalling or bettering the best solution found by the control strategy. It appears to produce more consistent results than the control strategy, although it remains to be shown whether this has any significant effect on its performance. As the number of cities exceeds 100, AEAC's performance becomes poorer. It is possible that this is because the number of links increases with the square of the number of cities. Hence, as the number of cities increases, the proportion of links which have been incorporated into solutions decreases, thereby reducing the amount of information available to calculate meaningful values of w. This may have detrimental effects on larger problems, as those links that have been used attract a weighting while most others do not, unfairly biasing the search away from some links and towards others. Future work could investigate ways of estimating the value of w for those links that have not yet been used. This should improve the performance of AEAC on larger problems by making the comparisons between used and unused links fairer. Another technique for gathering the required information is to use a greater number of ants, although this would have a negative impact on the algorithm's computational requirements.

An important limitation of the AEAC described is that experience is only accumulated over a single iteration and then discarded. At the beginning of each iteration none of the links have been chosen and so there is no information available to adjust their weightings. It is likely that this limitation greatly restricts the performance of AEAC. Future work will involve accumulated experience strategies that use experience gathered over *all* iterations. One such scheme is to keep track of the mean cost of complete solutions that have made use of each link. This is an extended form of the AEAC described here and may be better able to make objective evaluations of the utility of each link. Furthermore, if the cause of AEAC's poor performance on large problems is due to the lack of

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.
 https://doi.org/10.1142/S1469026803000938

accumulated knowledge on a large proportion of links, then this scheme may help to overcome the problem. Over the course of many iterations, information will be gathered about many more links than those used in just one iteration. We would expect this to lead to improved performance in our AEAC.

The AEAC is part of a wider strategy that is looking at ways of producing generic strategies to enhance ant based meta-heuristics [4]. Our next step is to determine how well AEAC applies across a range of problems, particularly those in which there is no obvious relation between adjacent elements.

References

1. Dorigo, M. and Gambardella, L. (1997) "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE Transactions on Evolutionary Computing, 1, pp. 53-66.
2. Dorigo, M. and Di Caro, G. (1999) "The Ant Colony Optimization Meta-heuristic", in New Ideas in Optimization, Corne, D., Dorigo, M. and Golver, F. (eds), McGraw-Hill, London, pp. 11-32.
3. Heusse, M., Gurin, S., Snyders, D., and Kuntz, P. (1998). "Adaptive agent-driven routing and load balancing in communications networks", Advances in Complex Systems, 1, pp. 237-254.
4. Randall, M. (2001) "A General Framework for Constructive Meta-heuristics", Operations Research/Management Science at Work: Applying Theory in the Asia Pacific Region, Springer Verlag, Berlin (in press).
5. Reinelt, G. (1991) "TSPLIB - A Traveling Salesman Problem Library", ORSA Journal of Computing, 3, pp. 376-384.

Electronic version of an article published at International Journal of Computational Intelligence and Applications, 3(2), 189-198.