points of the article

ANT COLONY SYSTEM - A COOPERATIVE LEARNING APPROACH TO THE TRAVELING SALESMAN PROBLEM

by M. Dorigo and L. Gambardella

1 INTRODUCTION

Motivation - Ants are capable of finding the shortest path by using Pheromone.
The algorithm of Ant Colony System (ACS) builds on the Ant System (AS) in which a set of artificiale ants cooperate to the solution of a problem by exchanging information via Pheromone deposit on graph edges.

Main idea - Set of simple agents, called ants, search in parallel for good solutions to the Traveling salesman problem (TSP) and cooperate through pheromone-mediated, indirect and global communication.

Informally, each ant construct a TSP solution in an iterative way : it adds new cities to a partial solution by exploiting both information gained from past experience (memory - Pheromone on edges) and a greedy heuristic (edges length).

2 BACKGROUND

2.1 Traveling salesman problem definitions

TSP

Let V = (a, ..., z) be a set of cities, A = {(r,s): r,s ϵ V} be the edge set, and δ(r,s)=δ(s,r) be a cost measure associated with edge (r,s) ϵ A.
the TSP is the problem of finding a minimal cost closed tour that visits each city once.

ATSP

If δ(r,s) $\neq$ δ(s,r) for at least some (r,s) then the TSP becomes an asymmetric TSP.

2.2 Ant system(AS) - the progenitor of ACS

In addition to the cost measure δ(r,s), each edge (r,s) has also a desirability measure τ(r,s), called Pheromone, which is updated at run time by artificial ants.

Informally, each ant generates a complete tour by choosing the cities according to a probabilistic State transation rule. Ants prefer to move to cities which are connected by short edges with a high amount of pheromone. Once all ants have completed their tours a Global Pheromone updating rule is applied; A fraction of the Pheromone evaporates on all edges and then each ant deposit an amount of Pheromone on edges which belong to its tour in proportion to how short its tour was. The process is then iterate.

The State transition rule(Random-propotional rule):

$$P_k(r,s) = \begin{cases} \frac{[\tau(r,s)][\eta(r,s)]^\beta}{\sum_{u \epsilon J_k(r)} [\tau(r,u)][\eta(r,u)]^\beta} & \text{if } s \epsilon J_k(r) \\ 0 & \text{otherwise} \end{cases}$$

$P_k(r,s)$ - probability with which ant k in city r chooses to move to city s.
$\tau(r,s)$ - Pheromone on edge (r,s).
$\eta(r,s)$ - 1/δ(r,s).
$J_k(r)$ - set of cities that remain to be visited by ant k positioned on city r.
β - parameter which determines the relative importance of Pheromone versus distance ($\beta > 0$).

The Global updating rule : Once all ants have built their tours, Pheromone is updates on all edges according to:

$\tau(r,s) \leftarrow (1-\alpha)\tau(r,s) + \sum_{k=1}^{m} \Delta\tau_k(r,s)$
where

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_k} & \text{if (r,s) } \epsilon \text{ tour done by ant k} \\ 0 & \text{otherwise} \end{cases}$$

$0 < \alpha < 1$ is the Pheromone decay parameter.
L_k is the length of the tour performed by ant k.
m is the number of ants.

Pheromone placed on the edges plays the role of a distributed long term memory. This allows an indirect form of communication called Stigmergy.

AS was useful for discovering good or optimal solutions for small TSPs (up to 30 cities), the time required to find such results made it infeasible for larger problems.

3 ACS

Informally, m ants are initially positioned on n cities chosen according to some initialization rule. Each ant builds a tour (feasible solution to the TSP) by repeatedly applying a stochastic greedy rule (the state transition rule). While constructing its tour, an ant also modifies the amount of Pheromone on the visited edges by applying the Local updating rule. Once all ants have terminated their tour, the amount of Pheromone on edges is modified again (by applying the Global updating rule). Three main changes to improve AS performance:

1. ACS State transition rule:

an ant positioned on node r chooses the city s to move to by applying

$$s = \begin{cases} max_{u \epsilon J_k(r)} \{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\} & \text{if } q < q_0 \\ S & \text{otherwise} \end{cases}$$

where

q is a random number uniformly distributed in [0..1].

q_0 is a parameter $(0 \leq q_0 \leq 1)$.

S is a random variable selected according to the probability distribution given in the AS's State transition rule.

This rule favors transition toward nodes connected by short edges with a large amount of Pheromone. The parameter q_0 determines the relative importance of exploitation (if $q \leq q_o$) versus biased exploration(otherwise).

2. ACS Global updating tule:

In ACS only the globally best ant (The ant which constructed the shortest tour from the beginning of the trial) is allowed to deposit Pheromone. Global updating is performed after all ants have completed their tours. The rule:

$$\tau(r,s) \leftarrow (1-\alpha)\tau(r,s) + \alpha\Delta\tau(r,s)$$

where

$$\Delta\tau(r,s) = \begin{cases} \frac{1}{L_{gb}} & \text{if (r,s) } \epsilon \text{ global-best-tour} \\ 0 & \text{otherwise} \end{cases}$$

$0 < \alpha < 1$ is the Pheromone decay parameter.

L_{gb} is length of the globally best tour from the beginning of the trial.

Experiments using L_{ib}(length of the best tour in the current iteration of the trial) instead of L_{gb}, have shown that difference between the two schemes is minimal, with a slight preference for global-best.

3. ACS local updating rule:

While building a solution to the TSP, ants visit edges and change their Pheromone level by applying the Local updating rule of :
$\tau(r, s) \leftarrow (1-\rho)\tau(r, s) + \rho\Delta\tau(r, s)$
where $0 < \rho < 1$ is a parameter.

Three values for the term $\Delta\tau(r, s)$ were experimented:
(a) Inspired by Q-learning, an algorithm developed to solve reinforcement learning problems : $\Delta\tau(r, s) = \gamma max_{z\epsilon J_k(s)}\tau(s, z)$ ($0 \leq \gamma < 1$ is a parameter)
(b) $\Delta\tau(r, s) = \tau_0$
(c) $\Delta\tau(r, s) = 0$

Table 1: *A comparison of local updating rules*

	ACS			Ant-Q			Acs with $\Delta\tau(r, s) = 0$			ACS without local-updating		
	AVG	SD	BEST	AVG	SD	BEST	AVG	SD	BEST	AVG	SD	BEST
City Set 1	**5.88**	0.05	**5.84**	**5.88**	0.05	**5.84**	5.97	0.09	5.85	5.96	0.09	**5.84**
City Set 2	**6.05**	0.03	**5.99**	6.07	0.07	**5.99**	6.13	0.08	6.05	6.15	0.09	6.05
City Set 3	**5.58**	0.01	**5.57**	5.59	0.05	**5.57**	5.72	0.12	**5.57**	5.68	0.14	**5.57**
City Set 4	**5.74**	0.03	**5.70**	5.75	0.04	**5.70**	5.83	0.12	**5.70**	5.79	0.05	5.71
City Set 5	**6.18**	0.01	**6.17**	**6.18**	0.01	**6.17**	6.29	0.11	**6.17**	6.27	0.09	**6.17**
Oliver30	424.74	2.83	**423.74**	**424.70**	2.00	**423.74**	427.52	5.21	**423.74**	427.31	3.63	423.91
ry48p	**14625**	142	**14422**	14766	240	**14422**	15196	233	14734	15308	241	14796

Table 1 shows that (a) (Ant-Q) and (b) (ACS) are best. Since ACS requires less computation that Ant-Q, the ACS was chosen to be used.

The effect of local updating is to make the desirability of edges change dynamically : every time an ant uses an edge this becomes slightly less desirable. In this way ants will make better use of Pheromone information -

without local updating all ants whould search in a narrow neighborhood of the best previous tour.

ACS parameter setting: The parameters are set to the following values - $\beta = 2, q_0 = 0.9, \alpha = \rho = 0.1, \tau_0 = (n \cdot L_{nn})^{-1}$
L_{nn} is the tour length produce by the nearest neighbor heuristic, n is the number of cities.

These values were obtained by a preliminary optimization phase, in which we found that the experimented optimal values were largly independent of the problem (except for τ_0).
The number of ants used is m=10 (according to experimental observation) and they are placed randomly, with at most one ant in each city.

4 ACS : SOME COMPUTATIONAL RESULTS

4.1 Comparison with other heuristics

First set of TSP problems comprises five randomly generated 50-city problems.

Table 2: *Comparison of ACS with other heuristics on TSP problems*

Problem name	ACS	SA	EN	SOM
City set 1	**5.88**	**5.88**	5.98	6.06
City set 2	6.05	**6.01**	6.03	6.25
City set 3	**5.58**	5.65	5.70	5.83
City set 4	**5.74**	5.81	5.86	5.87
City set 5	**6.18**	6.33	6.49	6.70

SA - simulated annealing
EN - elastic net
SOM - self-organizing map

ACS was run for 2500 iterations using 10 ants. ACS results are averaged over 25 trials.

Second set is composed of three geometric problems (taken from the real world) of between 50 and 100 cities.

Table 3: *Comparison of ACS with other heuristics on ATSP problems*

Problem name	ACS	GA	EP	SA	Optimum
Eil50	**425**	428	426	443	**425**
	(427.96)	(N/A)	**(427.86)**	(N/A)	(N/A)
	[**1830**]	[25000]	[100000]	[68512]	
Eil75	**535**	545	542	580	**535**
	(542.37)	(N/A)	(549.18)	(N/A)	(N/A)
	[**3480**]	[80000]	[325000]	[173250]	
Eil100	**21282**	21761	N/A	N/A	**21282**
	(21285.44)	(N/A)	(N/A)	(N/A)	(N/A)
	[**4820**]	[103000]	[N/A]	[N/A]	

GA - genetic algorithm
EP - evolutionary programming
SA - simulated annealing

ACS is run for 1250 iterations using 20 ants. ACS results are averaged over 15 trials.

It is important to test the ACS on both random and geometric instances of the TSP because these two calsses of problems have structural difference that can make them difficult for a particular algorithm and at the same time easy for another one.

4.2 ACS on some bigger problems

For bigger TSP problems it's common practice to use a data structure knows as a candidate list - static data structure which contain, for a given city i, the *cl* closest cities, ordered by increasing distances. We set *cl* to be 15.

Extended version of the ACS : an ant first chooses the city to move to among those belonging to the candidate list. Only if none of the cities in the list can be visited then it considers the rest of the cities.

Table 4: *ACS performance for some bigger geometric problems*

Problem name	BEST	TOURS	AVG	SD	OPTIMUM	ERROR (best)
d198	15888	585000	16054	71	15780	0.68%
pcb442	51268	595000	51690	188	50779	0.96%
att532	28147	830658	28523	275	27686	1.67%
rat783	9015	991276	9066	28	8806	2.37%
fl1577	22977	942000	23163	116	22230	3.37%

5 ACS PLUS LOCAL SEARCH

In past years, few ad-hoc heuristics for TSP have developed, which greatly outperform on the specific problem of the TSP, the general pupose algorithms approaches that were shown.

A general approach is to use tour constructive heuristics (as ACS) to generate a solution and then to apply a tour improvement heuristic to locally optimize it.

ACS-3-OPT : ACS with a tour improvement heuristic based on a restricted 3-OPT procedure in which three edges are exchanged.

ACS-3-OPT also uses candidate lists in its constructive part. If there is no feasible node in the list it chooses the closest node (this is different from ACS where in case the list contain no feasible nodes, then any of the remaining feasible cities can be chosen with a probability which is given by the normalized product of Pheromone and closeness).

5.1 Experimental results

Parameter setting :
$m = 10, \beta = 2, q_0 = 0.98, \alpha = \rho = 0.1, \tau_0 = (n \cdot L_{nn})^{-}1, cl = 20.$
For each test problem, ten trials have been executes.

ATSP problem :

Set of ATSP problems proposed in the first international contest on evolutionary optimization.

Results obtaind by ACS-3-OPT are compared with those obtained by ATSP-GA, the winner of the ATSP cometition.

STSP problem :

STSP-GA, the algorithm that won the first international contest on evolutionary optimization in the symetric TSP category.

ACS-3-OPT results are not as good as STSP-GA, but comperable to those obtained by other algorithms considered to be very good.

Table 5: *Comparison between ASC-3-Opt and ATSP-GA*

Problem name	Optimum	ACS len	ACS sec	ACS err	ATSP len	ATSP sec	ATSP err
p43	2810	2810	2	0.00%	2810	10	0.00%
ry48p	14422	14422	19	0.00%	14440	30	0.12%
ft70	38673	38679.8	6	0.02%	38683.8	639	0.03%
kro124p	36230	36230	25	0.00%	36235.3	115	0.01%
tfv170	2755	2755	68	0.00%	2766.1	211	0.40%

Table 6: *Comparison between ASC-3-Opt and STSP-GA*

Problem name	Optimum	ACS len	ACS sec	ACS err	STSP len	STSP sec	STSP err
d198	15780	15781.7	238	0.01%	15780	253	0.00%
lin318	42029	42029	537	0.00%	42029	2054	0.00%
att532	27686	27718.2	810	0.11%	27693.7	11780	0.03%
rat783	8806	8837.9	1280	0.36%	8807.3	21210	0.01%

A fair comparison of our results with those obtained with the currently best performing algorithms for symmetric TSP's is difficult since they use as local search a Lin-Kernighan heauristic based on a segment-tree data structure that is faster and gives better results than our restricted 3-opt procedure. It will be the subject of future work to add such a procedure to the ACS.

6 DISCUSSION AND CONCLUSIONS

An intuitive explanation of how the ACS works : Once all the ants have generated a tour, the best ant deposit its Pheromone, defining in this way a "preferred tour" for search in the next iteration. Still, guided exploration together with the fact that local updating "eats" Pheromone away allow for the search of new, possibly better tours in the neighborhood of the previous best tour. So the ACS can be seen as a sort of guided parallel stochastic search in the neighborhood of the best tour.

Recently there has been growing interest in the application of ant colony algorithm to difficult combinatorial problems.

Number of ways in whoch the ant colony approach can be improved and/or changed :
1. Number of ants which should contribute to the global updating rule.
2. Allowing ants which produce very bad tours to substract Pheromone.
3. Replace the current parallel local updating of Pheromone by a sequential one.
4. Add to the ACS a more effective local optimizer.

Subject of ongoing research is to establish the class of problems that can be attacked by the ACS (combinatorial optimization).

The ACS is an interesting novel approach to parallel stochastic optimization of the TSP.

A combination of a constructive method which generates good starting solution with local search which takes these solutions to a local optimum seems to be the best strategy, and the ACS is a very good constructive heuristic to provide such starting solutions for local optimizers.