Ant colonies for the traveling salesman problem

TR/IRIDIA/1996-3
Université Libre de Bruxelles
Belgium

Marco Dorigo
IRIDIA, Université Libre de Bruxelles, Avenue Franklin Roosevelt 50, CP 194/6, 1050 Bruxelles, Belgium
mdorigo@ulb.ac.be, http://iridia.ulb.ac.be/dorigo/dorigo.html

Luca Maria Gambardella
IDSIA, Corso Elvezia 36, 6900 Lugano, Switzerland
luca@idsia.ch, http://www.idsia.ch/~luca

Abstract

We describe an artificial ant colony capable of solving the traveling salesman problem (TSP). Ants of the artificial colony are able to generate successively shorter feasible tours by using information accumulated in the form of a pheromone trail deposited on the edges of the TSP graph. Computer simulations demonstrate that the artificial ant colony is capable of generating good solutions to both symmetric and asymmetric instances of the TSP. The method is an example, like simulated annealing, neural networks, and evolutionary computation, of the successful use of a natural metaphor to design an optimization algorithm.

Keywords: Ant colony optimization; Computational intelligence; Artificial life; Adaptive behavior; Combinatorial optimization; Reinforcement learning

Accepted for publication in BioSystems, 1997. In press.

1. Introduction

Real ants are capable of finding the shortest path from a food source to the nest (Beckers, Deneubourg and Goss, 1992; Goss, Aron, Deneubourg and Pasteels, 1989) without using visual cues (Hölldobler and Wilson, 1990). Also, they are capable of adapting to changes in the environment, for example finding a new shortest path once the old one is no longer feasible due to a new obstacle (Beckers, Deneubourg and Goss, 1992; Goss, Aron, Deneubourg and Pasteels, 1989). Consider Fig.1A: ants are moving on a straight line that connects a food source to their nest. It is well known that the primary means for ants to form and maintain the line is a pheromone trail. Ants deposit a certain amount of pheromone while walking, and each ant probabilistically prefers to follow a direction rich in pheromone. This elementary behavior of real ants can be used to explain how they can find the shortest path that reconnects a broken line after the sudden appearance of an unexpected obstacle has interrupted the initial path (Fig.1B). In fact, once the obstacle has appeared, those ants which are just in front of the obstacle cannot continue to follow the pheromone trail and therefore they have to choose between turning right or left. In this situation we can expect half the ants to choose to turn right and the other half to turn left. A very similar situation can be found on the other side of the obstacle (Fig.1C). It is interesting to note that those ants which choose, by chance, the shorter path around the obstacle will more rapidly reconstitute the interrupted pheromone trail compared to those which choose the longer path. Thus, the shorter path will receive a greater amount of pheromone per time unit and in turn a larger number of ants will choose the shorter path. Due to this positive feedback (autocatalytic) process, all the ants will rapidly choose the shorter path (Fig.1D). The most interesting aspect of this autocatalytic process is that finding the shortest path around the obstacle seems to be an emergent property of the interaction between the obstacle shape and ants distributed behavior: Although all ants move at approximately the same speed and deposit a pheromone trail at approximately the same rate, it is a fact that it takes longer to contour obstacles on their longer side than on their shorter side which makes the pheromone trail accumulate quicker on the shorter side. It is the ants' preference for higher pheromone trail levels which makes this accumulation still quicker on the shorter path. We will now show how a similar process can be put to work in a simulated world inhabited by artificial ants that try to solve the traveling salesman problem.

The traveling salesman problem (TSP) is the problem of finding a shortest closed tour which visits all the cities in a given set. In this article we will restrict attention to TSPs in which cities are on a plane and a path (edge) exists between each pair of cities (i.e., the TSP graph is completely connected).

2. Artificial ants

In this work an artificial ant is an agent which moves from city to city on a TSP graph. It chooses the city to move to using a probabilistic function both of trail accumulated on edges and of a heuristic value, which was chosen here to be a function of the edges length. Artificial ants probabilistically prefer cities that are connected by edges with a lot of pheromone trail and which are close-by. Initially, m artificial ants are placed on randomly selected cities. At each time step they move to new cities and modify the pheromone trail on the edges used –this is termed *local trail updating*. When all the ants have completed a tour the ant that made the shortest tour modifies the edges belonging to its tour –termed *global trail updating*– by adding an amount of pheromone trail that is inversely proportional to the tour length.

These are three ideas from natural ant behavior that we have transferred to our artificial ant colony: *(i)* the preference for paths with a high pheromone level, *(ii)* the higher rate of growth of the amount of pheromone on shorter paths, and *(iii)* the trail mediated communication among ants. Artificial ants were also given a few capabilities which do not have a natural counterpart, but which have been observed to be well suited to the TSP application: artificial ants can determine how far away cities are, and they are endowed with a working memory M_k used to memorize cities already visited (the working memory is emptied at the beginning of each new tour, and is updated after each time step by adding the new visited city).

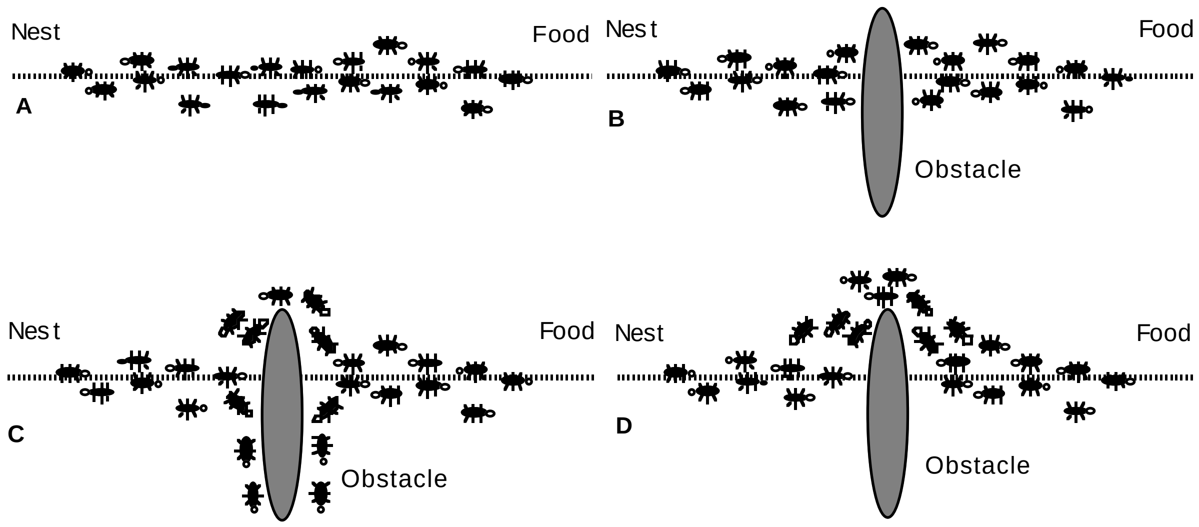

Fig.1. (A) Real ants follow a path between nest and food source. (B) An obstacle appears on the path: Ants choose whether to turn left or right with equal probability. (C) Pheromone is deposited more quickly on the shorter path. (D) All ants have chosen the shorter path.

There are many different ways to translate the above principles into a computational system apt to solve the TSP. In our *ant colony system* (ACS) an artificial ant *k* in city *r* chooses the city *s* to move to among those which do not belong to its working memory M_k by applying the following probabilistic formula:

$$s = \begin{cases} \arg\max_{u \notin M_k} \left\{ [\tau(r,u)] \cdot [\eta(r,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases} \tag{1}$$

where $\tau(r,u)$ is the amount of pheromone trail on edge (r,u), $\eta(r,u)$ is a heuristic function, which was chosen to be the inverse of the distance between cities *r* and *u*, β is a parameter which weighs the relative importance of pheromone trail and of closeness, *q* is a value chosen randomly with uniform probability in [0,1], q_0 ($0 \le q_0 \le 1$) is a parameter, and *S* is a random variable selected according to the following probability distribution, which favors edges which are shorter and have a higher level of pheromone trail:

$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)] \cdot [\eta(r,s)]^{\beta}}{\sum_{u \notin M_k} [\tau(r,u)] \cdot [\eta(r,u)]^{\beta}} & \text{if } s \notin M_k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $p_k(r,s)$ is the probability with which ant *k* chooses to move from city *r* to city *s*.

Pheromone trail is changed both *locally* and *globally*. Global updating is intended to reward edges belonging to shorter tours. Once artificial ants have completed their tours, the best ant deposits pheromone on visited edges; that is, on those edges that belong to its tour. (The other edges remain unchanged.) The amount of pheromone $\Delta\varphi(r,s)$ deposited on each visited edge (r,s) by the best ant is inversely proportional to the length of the tour: The shorter the tour the greater the amount of pheromone deposited on edges. This manner of depositing pheromone is intended to emulate the property of differential pheromone trail accumulation, which in the case of real ants was due to the interplay between length of the path and continuity of time. The *global trail updating* formula is $\varphi(r,s) \leftarrow (1-\alpha) \cdot \varphi(r,s) + \alpha \cdot \Delta\varphi(r,s)$, where $\Delta\varphi(r,s)$=(shortest tour)$^{-1}$. Global trail updating is similar to a reinforcement learning scheme in which better solutions get a higher reinforcement.

Local updating is intended to avoid a very strong edge being chosen by all the ants: Every time an edge is chosen by an ant its amount of pheromone is changed by applying the *local trail updating* formula: $\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s) + \alpha\cdot\tau_0$, where τ_0 is a parameter. Local trail updating is also motivated by trail evaporation in real ants.

Interestingly, we can interpret the ant colony as a reinforcement learning system, in which reinforcements modify the strength (i.e., pheromone trail) of connections between cities. In fact, the above formulas (1) and (2) dictate that an ant can either, with probability q_0, exploit the experience accumulated by the ant colony in the form of pheromone trail (pheromone trail will tend to grow on those edges which belong to short tours, making them more desirable), or, with probability (1-q_0), apply a biased exploration (exploration is biased towards short and high trail edges) of new paths by choosing the city to move to randomly, with a probability distribution that is a function of both the accumulated pheromone trail, the heuristic function, and the working memory M_k.

It is interesting to note that ACS employs a novel type of exploration strategy. First, there is the stochastic component *S* of formula (1): here the exploration of new paths is biased towards short and high trail edges. (Formula (1), which we call *pseudo-random-proportional* action choice rule, is strongly reminiscent of the pseudo-random action choice rule often used in reinforcement learning; see for example Q-learning (Watkins and Dayan, 1992)). Second, local trail updating tends to encourage exploration since each path taken has its pheromone value reduced by the local updating formula.

3. Results

We applied ACS to the symmetric and asymmetric TSPs listed in Tables 1, 2, 3, 4, and 7. These test problems were chosen either because there was data available in the literature to compare our results with those obtained by other naturally inspired methods or with the optimal solutions (the symmetric instances), or to show the ability of ACS in solving difficult instances of the TSP (the asymmetric instances).

Using the test problems listed in Tables 1, 2, and 3 the performance of ACS was compared with the performance of other naturally inspired global optimization methods: simulated annealing (SA), neural nets (NNs), here represented by the elastic net (EN) and by the self organizing map (SOM), evolutionary computation (EC), here represented by the genetic algorithm (GA) and by evolutionary programming (EP), and a combination of simulated annealing and genetic algorithms (AG); moreover we compared it with the farthest insertion (FI) heuristic. Numerical experiments were executed with ACS and FI, whereas the performance figures for the other algorithms were taken from the literature. The ACS parameters were set to the following values: m=10, β=2, α=0.1, q_0=0.9, $\tau_0=(n\cdot L_{nn})^{-1}$, where L_{nn} is the tour length produced by the nearest neighbor heuristic, and *n* is the number of cities (these values were found to be very robust across a wide variety of problems). In some experiments (see Table 2), the best solution found by the heuristic was carried to its local optimum by applying 3-opt (Croes, 1958). The tables show that ACS finds results which are at least as good as, and often better than, those found by the other methods. Also, the best solutions found by ACS in Table 2 were local optima with respect to 3-opt.

We also ran ACS on some bigger problems to study its behavior for increasing problem dimensions (see Table 4). For these runs we implemented a slightly modified version of ACS which incorporates a more advanced data structure known as *candidate list*, a data structure normally used when trying to solve big TSP problems (Reinelt, 1994; Johnson and McGeoch, in press). A candidate list is a list of preferred cities to be visited; it is a static data structure which contains, for a given city *i*, the *cl* closest cities. In practice, an ant in ACS with candidate list first chooses the city to move to among those belonging to the candidate list. Only if none of the cities in the candidate list can be visited then it considers the rest of the cities. In Tables 5 and 6 we study the performance of ACS for different lengths of the candidate list (ACS without candidate list corresponds to ACS with candidate list with the candidate list length set to *cl*=*n*). We report the results obtained for the Eil51 and Pcb442 TSPs (both these problem are included in TSPLIB) which show that a short candidate list improves both the average and the best performance of ACS; also, using a short candidate list it takes less CPU time to build a tour than using a longer one. The results reported in Table 4 were obtained setting *cl*=20.

Table 1. Comparison of ACS with other nature-inspired algorithms on random instances of the symmetric TSP. Comparisons on average tour length obtained on five 50-city problems. SA = simulated annealing, EN = elastic net, SOM = self organizing map, FI = farthest insertion. Results on SA, EN, and SOM are from (Durbin and Willshaw,1987; Potvin, 1993). FI results are averaged over 15 trials starting from different initial cities. ACS was run for 1,250 iterations using *m*=20 ants and the results are averaged over 15 trials. The best average tour length for each problem is in boldface.

Problem name	ACS	SA	EN	SOM	FI
City set 1	**5.86**	5.88	5.98	6.06	6.03
City set 2	6.05	**6.01**	6.03	6.25	6.28
City set 3	**5.57**	5.65	5.70	5.83	5.85
City set 4	**5.70**	5.81	5.86	5.87	5.96
City set 5	**6.17**	6.33	6.49	6.70	6.71

Table 2. Comparison of ACS with other nature-inspired algorithms on random instances of the symmetric TSP. Comparison on the shortest tour length obtained by SA+3-opt = best tour length found by simulated annealing and many distinct runs of 3-opt, SOM+ = best tour length found by SOM over 4,000 different runs (by processing the cities in various orders), FI, FI+3-opt = best tour length found by farthest insertion locally optimized by 3-opt, and ACS with and without local optimization by 3-opt. The 3-opt heuristics used the result of ACS and FI as starting configuration for local optimization. Results on SA+3-opt and SOM+ are from (Durbin and Willshaw,1987; Potvin, 1993). ACS was run for 1,250 iterations using *m*=20 ants and the best tour length was obtained out of 15 trials. The best tour length for each problem is in boldface.

Problem name	ACS	ACS + 3-opt	SA + 3-opt	SOM+	FI	FI + 3-opt
City set 1	**5.84**	**5.84**	**5.84**	**5.84**	5.89	5.85
City set 2	6.00	6.00	**5.99**	6.00	6.02	**5.99**
City set 3	**5.57**	**5.57**	**5.57**	5.58	**5.57**	**5.57**
City set 4	5.70	5.70	5.70	**5.60**	5.76	5.70
City set 5	**6.17**	**6.17**	**6.17**	6.19	6.50	6.40

Still more promising are the results we obtained applying ACS to some asymmetric TSP problems (Table 7). For example, ACS was able to find in 220 seconds (using a Pentium PC) the optimal solution for a 43-city asymmetric problem called 43X2. The same problem could not be solved to optimality within 32 hours of computation on a workstation by the best published code available for the asymmetric TSP based on the Assignment Problem relaxation (Fischetti and Toth, 1992) of the asymmetric TSP, and was only very recently solved to optimality by (Fischetti and Toth, 1994) with an algorithm based on polyhedral cuts (branch-and-cut scheme)[1].

[1] It should be noted that the task faced by ACS, as it is the case for any other heuristic method, is easier than that for any optimality proving algorithm since ACS does not have to prove the optimality of the obtained result. The results of Table 7 should therefore be taken for what they are: they suggest that ACS is a good method for finding good solutions to ATSPs in a reasonably short time, and not that ACS is competitive with exact methods.

Table 3. Comparison of ACS with the genetic algorithm (GA), evolutionary programming (EP), simulated annealing (SA), and the annealing-genetic algorithm (AG), a combination of genetic algorithm and simulated annealing (Lin, Kao and Hsu, 1993). We report the best integer tour length, the best real tour length (in parentheses) and the number of tours required to find the best integer tour length (in square brackets). Results using EP are from (Fogel, 1993) and those using GA are from (Bersini, Oury and Dorigo, 1995) for KroA100, and from (Whitley, Starkweather and Fuquay, 1989) for Oliver30, Eil50, and Eil75. Results using SA and AG are from (Lin, Kao and Hsu, 1993). Oliver30 is from (Oliver, Smith and Holland, 1987), Eil50, Eil75 are from (Eilon, Watson-Gandy and Christofides, 1969) and are included in TSPLIB[2] with an additional city as Eil51.tsp and Eil76.tsp. KroA100 is also in TSPLIB. The best result for each problem is in boldface. It is interesting to note that the complexity of all the algorithms is order of $n^2 \cdot t$, except for EP for which it is order of $n \cdot t$ (where n is the number of cities and t is the number of tours generated). It is therefore clear that ACS and EP greatly outperform GA, SA, and AG.

Problem name	ACS	GA	EP	SA	AG	Optimum
Oliver30	**420**	421	**420**	424	**420**	**420**
(30-city problem)	**(423.74)**	(N/A)	**(423.74)**	(N/A)	(N/A)	**(423.74)**
	[830]	[3,200]	[40,000]	[24,617]	[12,620]	
Eil50	**425**	428	426	443	436	**425**
(50-city problem)	(427.96)	(N/A)	**(427.86)**	(N/A)	(N/A)	(N/A)
	[1,830]	[25,000]	[100,000]	[68,512]	[28,111]	
Eil75	**535**	545	**542**	580	561	**535**
(75-city problem)	**(542.31)**	(N/A)	(549.18)	(N/A)	(N/A)	(N/A)
	[3,480]	[80,000]	[325,000]	[173,250]	[95,506]	
KroA100	**21,282**	21,761	N/A	N/A	N/A	**21,282**
(100-city problem)	**(21,285.44)**	(N/A)	(N/A)	(N/A)	(N/A)	(N/A)
	[4,820]	[103,000]	[N/A]	[N/A]	[N/A]	

Table 4. ACS Performance for some bigger TSPs. First column: best result obtained by ACS out of 15 trials; we give the integer length of the shortest tour, and the number of tours which were generated before finding it (in square brackets). Second column: ACS average on 15 trials and its standard deviation in square brackets. Third column: optimal result (for fl1577 we give, in square brackets, the known lower and upper bounds, given that the optimal solution is not known). Fourth column: error percentage, a measure of the quality of the best result found by ACS. Fifth column: time required to generate a tour on a Sun Sparc–server (50 MHz). The reason for the more than linear increase in time is that the number of failures, that is, the number of times an ant has to choose the next city outside of the candidate list, increases with the problem dimension. All problems are included in TSPLIB.

Problem name	ACS (*cl*=20) best result (1)	ACS average	Optimal result (2)	% Error $\frac{(1)-(2)}{(2)}$	CPU seconds to generate a tour
d198 (198-city problem)	15,888 [585,000]	16,054 [71.1]	15,780	0.68 %	0.02
pcb442 (442-city problem)	51,268 [595,000]	51,690 [188.7]	50,779	0.96 %	0.05
att532 (532-city problem)	28,147 [830,658]	28,522 [275.4]	27,686	1.67%	0.07
rat778 (778-city problem)	9,015 [991,276]	9,066 [28.2]	8,806	2.37 %	0.13
fl1577 (fl1577-city problem)	22,977 [942,000]	23,163 [116.6]	[22,137 – 22,249]	3.27÷3.79 %	0.48

2 TSPLIB: http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html (maintained by G. Reinelt).

Table 5. Comparison between candidate list size. Problem: Eil51. For each candidate list length, averages are computed over 15 trials. In each trial the number of tour generated is 500.

Candidate list length	ACS average	ACS best result	Average time per trial (sec)	Average number of failures for each tour built
10	431.00	426	13.93	0.73
20	431.27	427	23.93	0.48
30	435.27	429	33.93	0.36
40	433.47	426	44.26	0.11
50	433.87	429	55.06	0.01

Table 6. Comparison between candidate list size. Problem: Pcb442. For each candidate list length averages is computed over 10 trials. In each trial the number of tour generated is 20,000.

Candidate list length	ACS average	ACS best result	Average time per trial (sec)	Average number of failures for each tour built
20	54,024.9	52,201	458.5	3.42
40	54,970.9	53,580	786.4	2.10
60	55,582.7	53,907	1,134.5	1.77
80	56,495.9	54,559	1,459.2	1.53
100	56,728.3	54,527	1,764.3	1.30

Table 7. Comparison between exact methods and ACS for ATSP problems. The exact method is the best published deterministic code for the asymmetric TSP; results are from (Fischetti and Toth, 1992; 1994). The ry48p is from TSPLIB, and the 43X2 problem is from (Balas, Ceria and Cornuéjols, 1993). We report the tour length and, in parentheses, CPU seconds used to find the reported solution (experiments were run on a Pentium PC). ACS was run using 10 ants for 1500 iterations, and results were obtained out of 15 trials. The best result for each problem is in boldface.

Problem	ACS best result	ACS average	FT-92	FT-94
43X2 (43-city problem)	**5,620** **(220)**	5,627 (295)	N/A	**5,620** (492.2)
ry48p (48-city problem)	**14,422** (610)	14,685 (798)	**14,422** (729.6)	**14,422** **(52.8)**

In addition to providing interesting computational results, ACS presents also some attractive characteristics due to the use of trail mediated communication. First, communication determines a synergistic effect. This is shown for example in Fig.2, which shows the typical result of an experiment in which we studied the average speed to find the optimal solution (defined as the inverse of the average time to find the optimal solution) as a function of the number of ants in ACS. To make the comparison fair performance was measured by CPU time, so as to discount for the higher complexity of the algorithm when ants communicate (higher complexity is due to trail updating operations: when ants do not communicate trail is initially set to 1 on all edges and is not updated during computation). Second, communication increases the probability of finding quickly an optimal solution. Consider the distribution of the first finishing times, where the first finishing time is the time elapsed until the first optimal solution is found. Fig.3 shows how this distribution changes in the communicating and the noncommunicating cases. These results show that communication among ants (mediated by trail) is useful.

Although when applied to the symmetric TSP ACS is not competitive with specialized heuristic methods like Lin-Kernighan (Lin and Kernighan, 1973), its performance can become very interesting when applied to a slightly different problem; in this article we reported some results on the asymmetric TSP. An extended version of ACS has recently been applied to the quadratic assignment problem (Dorigo, Gambardella and Taillard, 1997): it was able to find solutions of a quality (measured as cost of the obtained result and as CPU time required to find it) comparable to that of solutions found by the currently best heuristics for the QAP: Taboo Search (Taillard, 1991), an hybrid genetic algorithm (Fleurent and Ferland, 1994), and GRASP (Li, Pardalos and Resende, 1994).

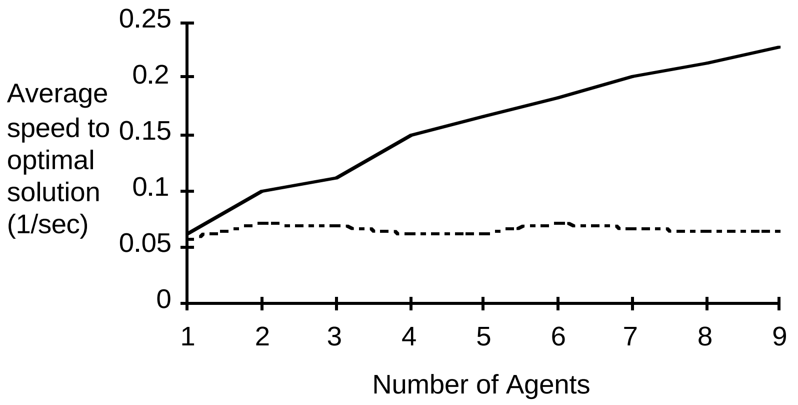

Fig.2. Communication determines a synergistic effect. Communication among agents: solid line. Absence of communication: dotted line. Test problem: CCAO, a 10-city problem (Golden and Stewart, 1985). Average on 100 runs. (The use of an increasing number of ants does not improve performance for the case of absence of cooperation since we use CPU time to measure speed.)

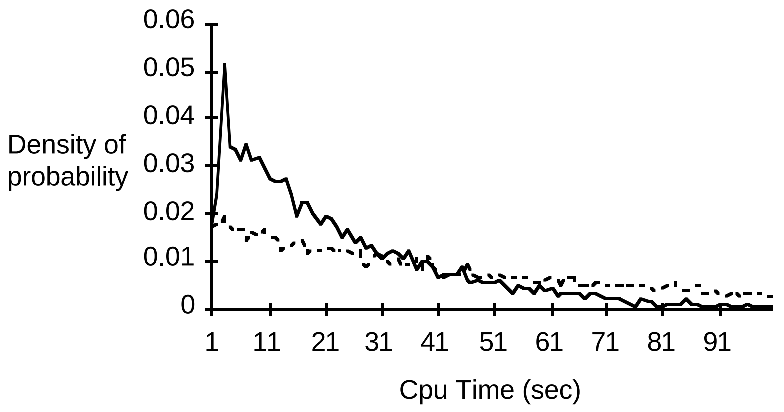

Fig.3. Communication changes the probability distribution of first finishing times. Communication among agents: solid line. Absence of communication: dotted line. Test problem: CCAO, a 10-city problem (Golden and Stewart, 1985). Average on 10,000 runs. Number of ants: *m*=4.

4. Conclusions

The key to the application of ACS to a new problem is to identify an appropriate representation for the problem (to be represented as a graph searched by many artificial ants), and an appropriate heuristic that defines the distance between any two nodes of the graph. Then the probabilistic interaction among the artificial ants mediated by the pheromone trail deposited on the graph edges will generate good, and often optimal, problem solutions.

There are many ways in which ACS can be improved so that the number of tours needed to reach a comparable performance level can diminish, making its application to larger problem instances feasible. First, a local optimization heuristic like 2-opt, 3-opt or Lin-Kernighan (Lin and Kernighan, 1973) can be embedded in the ACS algorithm (this is a standard approach to improve efficiency of general purpose algorithms like EC, SA, NNs, as discussed in (Johnson and McGeoch, in press)). In the experiments presented in this article, local optimization was just used to improve on the best results produced by the various algorithms. On the contrary, each ant could be taken to its local optimum before global trail updating is performed. Second, the algorithm is amenable to efficient parallelization, which could greatly improve the performance for finding good solutions, especially for high-dimensional problems. The most immediate parallelization of ACS can be achieved by distributing ants on different processors: the same TSP is then solved on each processor by a smaller number of ants, and the best tour found is exchanged asynchronously among processors. A preliminary implementation (Bolondi and Bondanza, 1993) of a similar scheme (Dorigo, Maniezzo and Colorni, 1996) on a net of transputers has shown that it can make the complexity of the algorithm largely independent of the number of ants. Third, the method is open to further improvements such as the introduction of specialized families of ants, tighter connections with reinforcement learning methods (Gambardella and Dorigo, 1995; Dorigo and Gambardella, 1996), and the introduction of more specialized heuristic functions to direct the search.

Acknowledgments

Marco Dorigo is a Research Associate with the FNRS. Luca Gambardella is Research Director at IDSIA. This research has been funded by the Swiss National Science Fund, contract 21–45653.95 titled “Cooperation and learning for combinatorial optimization”. We wish to thanks David Fogel and one anonymous referee for the many useful comments on a previous version of this article.

References

Balas, E., Ceria, S. and Cornuéjols, G., 1993, A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58, 295–324.

Beckers, R., Deneubourg, J.L. and Goss, S., 1992, Trails and U-turns in the selection of the shortest path by the ant Lasius niger. *Journal of Theoretical Biology*, 159, 397–415.

Bersini, H., Oury, C. and Dorigo, M., 1995, Hybridization of Genetic Algorithms. *Tech. Rep. No. IRIDIA 95-22*, IRIDIA, Université Libre de Bruxelles, Belgium.

Bolondi, M. and Bondanza, M., 1993, *Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore*. Master thesis, Politecnico di Milano, Italy.

Croes, G.A., 1958, A method for solving traveling salesman problems. *Operations Research*, 6, 791–812.

Dorigo, M. and Gambardella, L.M., 1996, A study of some properties of Ant-Q, in: *Proceedings of PPSN IV–Fourth International Conference on Parallel Problem Solving From Nature*, H.–M. Voigt, W. Ebeling, I. Rechenberg and H.–S. Schwefel (eds.) (Springer-Verlag, Berlin) pp. 656–665.

Dorigo, M., Maniezzo, V. and Colorni, A., 1996, The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26, 29–41.

Durbin, R. and Willshaw, D., 1987, An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326, 689-691.

Eilon, S., Watson-Gandy, C.D.T. and Christofides, N., 1969, Distribution management: mathematical modeling and practical analysis. *Operational Research Quarterly*, 20, 37–53.

Fischetti, M. and Toth, P., 1992, An Additive Bounding Procedure for the Asymmetric Travelling Salesman Problem. *Mathematical Programming*, 53, 173–197.

Fischetti, M. and Toth, P., 1994, A polyhedral approach for the exact solution of hard ATSP instances. *Tech. Rep. No. OR-94*, DEIS, Università di Bologna, Italy.

Fleurent, C. and Ferland, J., 1994, Genetic Hybrids for the Quadratic Assignment Problem, in: *DIMACS Series in Discrete Mathematics and Theorical Computer Science*, 16, pp. 173–187.

Fogel, D., 1993, Applying evolutionary programming to selected traveling salesman problems. *Cybernetics and Systems: An International Journal*, 24, 27–36.

Gambardella, L.M., Taillard, E., and Dorigo, M., 1997. Ant Colonies for QAP. *Tech. Rep. No. IDSIA 97-4*, IDSIA , Lugano, Switzerland.

Gambardella, L. M. and Dorigo, M., 1995, Ant-Q: A Reinforcement Learning approach to the traveling salesman problem, in: *Proceedings of ML-95, Twelfth International Conference on Machine Learning*, A. Prieditis and S. Russell (eds.) (Morgan Kaufmann, San Francisco, CA) pp. 252–260.

Golden, B. and Stewart, W., 1985, Empiric analysis of heuristics, in *The Traveling Salesman Problem*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan and D. B. Shmoys (eds.) (Wiley, New York) pp. 207–250.

Goss, S., Aron, S., Deneubourg, J.L. and Pasteels, J.M., 1989, Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76, 579–581.

Hölldobler, B. and Wilson, E.O., 1990, *The ants* (Springer-Verlag, Berlin).

Johnson, D.S. and McGeoch, L.A., in press. The Travelling Salesman Problem: A Case Study in Local Optimization, in: *Local Search in Combinatorial Optimization*, E.H.L. Aarts and J.K. Lenstra (eds.) (Wiley, New York).

Li, Y., Pardalos, P.M., and Resende, M.G.C., 1994, A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem, in: *Quadratic assignment and related problems*, P. M. Pardalos and H. Wolkowicz (eds.) (DIMACS Series, American Mathematical Society, Rhode Island) pp. 237-261.

Lin, F.–T., Kao, C.–Y. and Hsu, C.–C., 1993, Applying the genetic approach to simulated annealing in solving some NP-hard problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 23, 1752–1767.

Lin, S. and Kernighan, B.W., 1973, An effective Heuristic Algorithm for the TSP. *Operations Research*, 21, 498–516.

Oliver, I., Smith, D. and Holland, J.R., 1987, A study of permutation crossover operators on the travelling salesman problem, in: *Proceedings of the Second International Conference on Genetic Algorithms*, J.J. Grefenstette (ed.) (Lawrence Erlbaum, Hillsdale, New Jersey) pp. 224–230.

Potvin, J–Y., 1993, The traveling salesman problem: A neural network Perpective. *ORSA Journal of Computing*, 5 (4), 328–347.

Reinelt, G., 1994, *The traveling salesman: Computational solutions for TSP applications* (Springer-Verlag, Berlin).

Taillard, E., 1991, Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17, 443–455.

Watkins, C.J.C.H. and Dayan, P., 1992, Technical note: Q-learning. *Machine Learning*, 8, 279–292.

Whitley, D., Starkweather, T. and Fuquay, D., 1989, Scheduling problems and travelling salesman: the genetic edge recombination operator, in: *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.) (Morgan Kaufmann, San Mateo, CA) pp. 133–140.