Adv. Complex Systems (1998) **1**, 149–159

Evolving Ant Colony Optimization

Hozefa M. Botee
Santa Fe Institute
1399 Hyde Park Road
Santa Fe, NM 87501, USA
botee@santafe.edu

Eric Bonabeau†
Santa Fe Institute
1399 Hyde Park Road
Santa Fe, NM 87501, USA
bonabeau@santafe.edu

ABSTRACT. Ant Colony Optimization (ACO) is a promising new approach to combinatorial optimization. Here ACO is applied to the traveling salesman problem (TSP). Using a genetic algorithm (GA) to find the best set of parameters, we demonstrate the good performance of ACO in finding good solutions to the TSP.
KEYWORDS: Combinatorial Optimization; Traveling Salesman Problem; Heuristics; Ant Colony Optimization; Genetic Algorithm.

1. Introduction

Social insects—ants, bees, termites and wasps—exhibit a collective problem-solving ability (Deneubourg and Goss, 1989; Bonabeau *et al.*, 1997). In particular, several ant species are capable of selecting the shortest pathway, among a set of alternative pathways, from their nest to a food source (Beckers *et al.*, 1990). Ants deploy a chemical trail (or *pheromone* trail) as they walk; this trail attracts other ants to take the path that has the most pheromone. This reinforcement process results in the selection of the shortest path: the first ants coming back to the nest are those that took the shortest path twice (to go from the nest to the source and to return to the nest), so that more pheromone is present on the shortest path than on longer paths immediatly after these ants have returned, stimulating nestmates to choose the shortest path. Taking advantage of this ant-based optimizing principle combined with pheromone evaporation to avoid early convergence to bad solutions, Colorni *et al.* (1991, 1992, 1993), Dorigo *et al.* (1996), Dorigo and Gambardella (1997a, 1997b; Gambardella and Dorigo, 1995) and Gambardella *et al.* (1997) have proposed a remarkable optimization method,

† To whom correspondence should be sent.

Ant Colony Optimization (ACO), which they applied to classical NP-hard combinatorial optimization problems, such as the traveling salesman problem (Lawler *et al.*, 1985), the quadratic assignment problem (Gambardella *et al.*, 1998) or the job-shop scheduling problem (Colorni *et al.*, 1993), with reasonable success. More applications are described by Bonabeau *et al.* (1998). The parameters of the ACO algorithms developed in these papers were hand-tuned. In the present letter we demonstrate the good performance of ACO algorithms when parameters are selected using a systematic procedure. More precisely we use a genetic algorithm (Goldberg, 1989) to evolve ACO algorithms. A simple implementation of this approach, tested on the traveling salesman problem (TSP), resulted in: (1) increased convergence speed (compared to the performance of the best hand-tuned ACO algorithm) toward the optimal solution for a 30-city problem (Whitley *et al.*, 1989), and (2) several very good floating point solutions for a 51-city problem (Eilon *et al.*, 1969). Our results suggest that it might be possible to systematically find parameters that significantly increase the performance of ACO algorithms, and confirm that ACO is more than an exotic metaheuristic as it compares well with existing algorithms on popular benchmark problems.

2. ACO algorithm for the TSP

In this section we briefly describe Ant Colony System (ACS), an ACO algorithm introduced by Dorigo and Gambardella (1997a, 1997b). Let us consider a symmetric TSP with n cities. Let m be the total number of ants, assumed constant over time. For an ant located on city i, the transition from city i to city j depends on:

(1) Whether or not city j has already been visited. Each ant has a tabu list that contains all the cities that the ant has already visited. Let $J_k(i)$ be the set of cities that remain to be visited by ant k when ant k is currently on city i.

(2) The distance d_{ij} between i and j. $d_{ij} = d_{ji}$ for a symmetric TSP.

(3) The amount of "artificial pheromone" on the edge connecting i to j, denoted by (i, j).

Let $\tau_{ij}(t)$ be the total amount of pheromone on edge (i, j) at time t. Time is incremented by 1 when all ants have completed a tour. The initial amount of pheromone on edges is assumed to be a small positive constant c: $\forall(i, j)$, $\tau_{ij}(t = 0) = c$. At the beginning of each iteration, ants are placed randomly on the cities. When on city i, ant k selects which city, j, to move to. To do so ant k checks the candidate list associated with city i, which is a list of preferred cities to be visited from city i: instead of examining all possibilities from any given city, unvisited cities in the candidate list are examined first, and only when all cities in the candidate list have been visited are other cities examined. The candidate list of a city contains the n_l closest cities. Cities in the candidate list are ordered by increasing distance, and the list is scanned sequentially. Ant k first chooses the next city to hop to from the list, and, if all cities in the candidate list have already been visited, selects city j according to:

$$j = \begin{cases} \underset{u \in J_k(i)}{\text{Arg Max}} \left\{ [\tau_{iu}]^{\alpha} \cdot [d_{iu}]^{-\beta} \right\} & \text{if } q \leq q_0, \\ J & \text{otherwise.} \end{cases} \tag{2.1}$$

where q is a real random variable uniformly distributed in the interval $[0,1]$, q_0 is a tunable parameter ($0 \leq q_0 \leq 1$), and $J \in J_k(i)$ is a node that is randomly selected according to probability

$$p_{iJ}^{k}(t) = \frac{[\tau_{iJ}(t)]^{\alpha} \cdot [d_{iJ}]^{-\beta}}{\sum\limits_{u \in J_k(i)} [\tau_{iu}(t)]^{\alpha} \cdot [d_{iu}]^{-\beta}} \tag{2.2}$$

where α and β are two adjustable parameters that control the relative influences of trail intensity ($\tau_{iJ}(t)$) and distance (d_{iJ}). If α=0, the closest cities are more likely to be selected: this corresponds to a classical stochastic greedy algorithm with multiple starting points because ants are initially randomly distributed on the nodes. If β=0, only pheromone amplification is at work: this method will lead to the rapid selection of tours which may be far from optimal. $q \leq q_0$ corresponds to an exploitation of the network, that makes use of distances between cities and of existing pheromone trails by choosing the best local compromise between distance and pheromone concentration, whereas $q > q_0$ favors more exploration. Cutting exploration by tuning q_0 allows to concentrate the activity of the system on the best solutions, instead of letting it explore constantly.

Pheromone trails are updated locally and globally:
Local update. When, while performing a tour, ant k is on city i and selects city j ($\in J_k(i)$) as the next city to hop to, the pheromone concentration of (i,j) is immediatly reinforced by a fixed amount τ_0. The trail decays simultaneously, so that:

$$\tau_{ij} \leftarrow (1-\rho_l) \cdot \tau_{ij} + \rho_l \cdot \tau_0 \tag{2.3}$$

where ρ_l ($0 \leq \rho_l \leq 1$) is a parameter governing local trail decay.
Global update. The ant that performed the best tour since the beginning of the trial is allowed to globally update the concentrations of pheromone on the corresponding edges. To improve the solutions found by ants, a local search procedure is performed. In the case of TSP, the most widespread local search procedures are $2-opt$ and $3-opt$ (Lin, 1965), and Lin-Kernighan (LK) (Lin and Kernighan, 1973), whereby two, three and a variable number of edges, respectively, are exchanged. In the present paper we restrict ourselves to $2-opt$. Each ant's tour is modified by applying $2-opt$ a certain number of times, denoted by σ. If, after application of the sequence of σ $2-opt$ swaps, no better solution has been found the original tour is kept, otherwise the better tour is kept. $\tau_{ij}(t)$ is then modified by an amount $\Delta\tau_{ij}(t)$ as follows:

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{Q}{(L_+)^{\gamma}} & \text{if } (i,j) \in T_+, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

where Q is a tunable parameter, T_+ is the best tour since the beginning of the trial and L_+ is its length. $\gamma = 1$ is a parameter introduced in this study. Only the best tour is reinforced through the global update. Here again, trail decay is implemented:

$$\tau_{ij}(t+1) \leftarrow (1-\rho_g) \cdot \tau_{ij}(t) + \rho_g \cdot \Delta\tau_{ij}(t) \tag{2.5}$$

where ρ_g $(0 \leq \rho_g \leq 1)$ is a paremeter governing global trail decay.

Using this method with hand-tuned parameters, Dorigo and Gambardella (1997a, 1997b) obtained very good results, in terms of the quality of the solutions generated and of CPU time. Using a more systematic method to determine parameters we report an even better performance in the next section.

3. Evolving ACO for the TSP

3.1. Parameters evolved

We use a GA to evolve some of the parameters of the ACO algorithm described in the previous section. The parameters that have been subjected to the GA search are summarized in Table 1 together with their ranges of variation. The number of ants, m, is included because it is unclear how many ants are necessary to find a very good solution in an efficient way for a problem of a given size. A related question is: how does the optimal value of m scale with n? The exploitation probability, q_0, is another parameter for which the optimal value is unclear. Intuitively it seems this parameter should be relatively problem-independent. The two parameters α and β, which determine the respective weights of the pheromone trail and the distance between two nodes, and the two parameters ρ_l and ρ_g, which govern local and global trail decay, respectively, are also important and, hopefully, are problem-independent. The same may apply to τ_0, the amount of local reinforcement, although Dorigo and Gambardella (1997a, 1997b) suggest that it depends on the typical length of a tour obtained from a greedy algorithm. The Q parameter scales the amount by which the trail is modified in the global update procedure. The optimal value of Q may or may not be problem-dependent. The same holds for γ, which, depending on its value, amplifies or reduces the influence of shorter tours on the global update. The number of $2-opt$ swaps, σ, is assumed to be of the form $\sigma = E[a \cdot n^b]$, where n is the number of cities, a and b are two positive parameters which determine how σ should scale with problem size, and $E[x]$ is the integer part of x. a and b are part of the evolutionary search. Finally, in principle the size of the candidate list, n_l, should also be included in the search. Since we consider relatively small problems here, we set $n_l = 1$, that is, the candidate list contains only the closest city. It will be interesting in future studies to investigate how the optimal value of n_l scales with problem size.

3.2. Fitness function and GA

In the GA, each colony, characterized by a set of parameters, is an individual. Colonies are in competition to make it to the next generation. The fitness func-

Table 1: Parameters evolved and their ranges.

Range
$1 \leq m \leq 2 \cdot n$
$0.0 \leq q_0 \leq 1.0$
$0.0 \leq \alpha \leq 5.0$
$0.0 \leq \beta \leq 10.0$
$0.0 \leq \rho_{local} \leq 1.0$
$0.0 \leq \rho_{global} \leq 1.0$
$0.0 \leq Q \leq 100.0$
$0.1 \leq \gamma \leq 3.0$
$0.0 \leq \tau_0 \leq 0.5$
$0 \leq a \leq 9$
$0.0 \leq b \leq 1.0$
$n_l = 1$

tion F used in this study is the weighted sum of four components: $F = \sum_{i=1}^{4} c_i \cdot F_i$, where c_i is the weight of component i.

(1) $F_1 = \frac{1}{L+1-L_+}$, where L is the best tour found by the colony and L_+ is the best tour length found by all of the colonies thus far. Although in principle this component can diverge we never encountered a problem in our simulations. Note that this component is relative to the performance of other colonies. $c_1 = 2.0 - 3.0$.

(2) $F_2 = e^{-\frac{v}{5 \cdot n}}$, where v is the iteration in which the best tour was found. F_2 reflects the fact that it is important for the best tour to be found quickly. $c_2 = 0.5 - 0.8$.

(3) $F_3 = e^{-\frac{m}{10 \cdot n}}$ encourages m to be as small as possible. It is important to try to minimize m as the CPU time per iteration scales linearly with m. $c_3 = 0.5$.

(4) $F_4 = e^{-\frac{\sigma}{n}}$. It is important to minimize the number of $2-opt$ swaps per ant as the CPU time per iteration scales linearly with σ. $c_4 = 0.2 - 0.5$.

Note that F_3 and F_4 do not directly depend on how well the algorithm performs but rather on parameters that influence its speed of convergence toward a good solution. Alternatively, and better even, components F_2, F_3, and F_4 could be replaced by the CPU time required to find the best tour.

We use a simple GA implementation, the details of which are described by Riolo (1992). A random genome is created for each individual with each gene as a bit string of length 16 coding for a parameter. The fitness of each individual is evaluated by running the ACS algorithm. Selection, crossover and mutation take place according to the specifications in Riolo (1992). The percentages chosen for selection and crossover are both 75% and the rate of mutation is 0.3%. All runs have a population size of 40 and stop after 100 generations.

3.3. Results

3.3.1. Oliver30

The description of this Euclidian 30-city TSP can be found in Whitley *et al.* (1989).

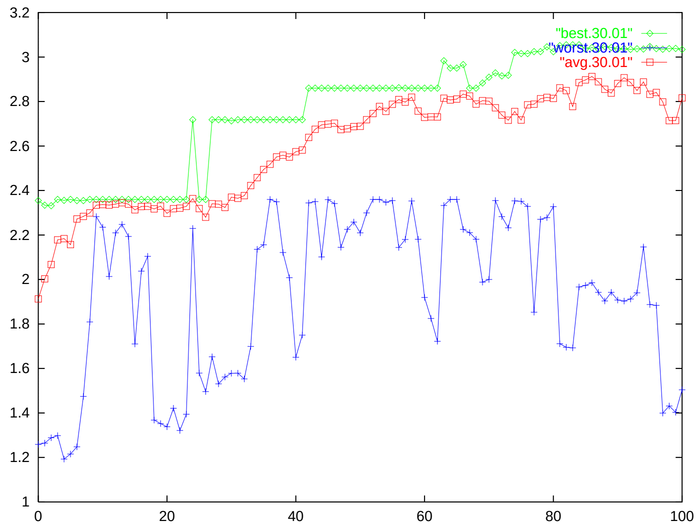

Figure 1: Best, worst and average fitness as a function of generation number for Oliver30.

Table 2: Parameters for Oliver30.

Parameter	Value
m	14
q_0	0.384665
α	0.370642
β	6.683146
ρ_{local}	0.302831
ρ_{global}	0.308385
Q	78.042267
γ	0.671618
τ_0	0.409461
a	5
b	0.968872

Figure 1 shows the evolution of the best, worst and average fitness. The evolutionary search was able to find parameter values (summarized in Table 2) which significantly improved on the results obtained by Dorigo and Gambardella (1997a,b). These authors describe a set of parameters which allows ACS to find the optimal solution (represented in Figure 2, integer length: 420, floating point length: 423.74) after 830 ACS iterations with 10 ants, that is, after 8300 ant-cycles. One ant-cycle corresponds to one ant performing a complete tour. An algorithm, with 14 ants, which always finds the optimal solution was found with the GA: the algorithm was tested 30 times and the average number of cycles it requires to find the optimal solution is 352 (std=260.53), that is, 4928 ant-cycles.

3.3.2. Eil51

Eil51 can be obtained in the TSPLIB library of problems for the TSP at: ftp://ftp.zib.de/pub/Packages/mp-testdata/tsp/tsplib/index.html.

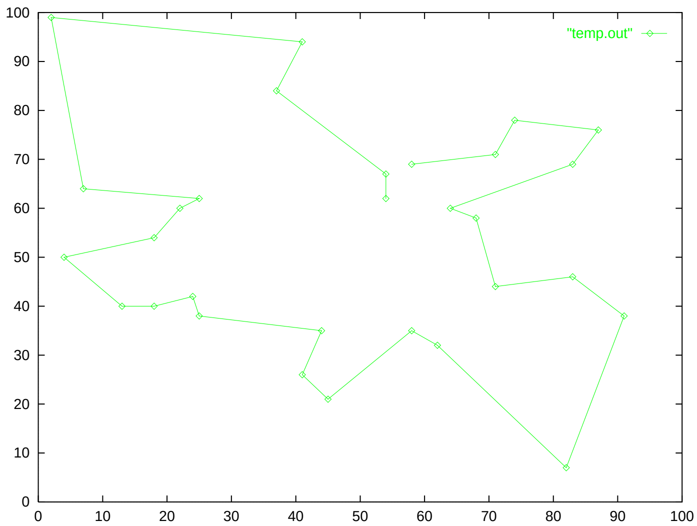

Figure 2: Optimal tour for Oliver30, Length = 423.740563

The best known integer length solution to Eil51, listed in TSPLIB, is represented in Figure 3 and has a floating-point length of 429.983312. The parameters obtained by running the GA on Oliver30 were used (Table 2) with the exception of the number of ants, m, which was set to $m = 25$. The parameters obtained for Oliver30 were assumed to work well for Eil51 as these two problems are characterized by similar optimal tour lengths. This assumption is confirmed by the quality of the tours found by the algorithm. We were able to find three tours with respective floating-point lengths 429.737129 (Figure 3.3.2), 429.117939 (Figure 5) and 428.981647 (Figure 6). The details of these tours are given in the figure legends. While Figures 3.3.2 and 5 represent tours that are variations on the best known tour, Figure 6 represents a solution which varies significantly from the other solutions.

4. Conclusion and future work

The results presented in this paper are extremely encouraging. They indicate that using an evolutionary search to find the best parameters of an ACO algorithm can result in significant computation time improvments on existing ACO algorithms. The obtained algorithms compare well with other existing combinatorial optimization algorithms on Eil51, suggesting that it could also be true for other TSP instances as well as for other problems. Maniezzo and Colorni (1998) report that hand-tuned ACO algorithms applied to the quadratic assignment problem improve on the best known results on structured problems, that is, instances taken from real-world applications: in view of the results of the present paper, tuning the parameters of the ACO algorithms with an automated search might result in even better solutions. We have to mention for completeness the work of White *et al.* (1998), who seem to have been the first to actually use a

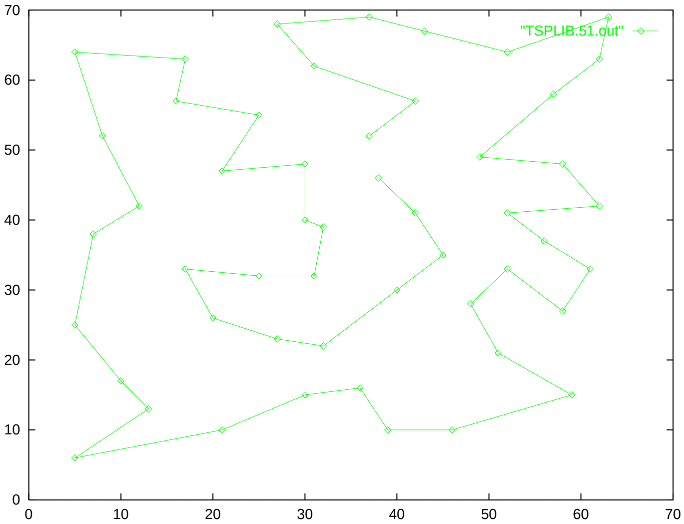

Figure 3: Best tour for Eil51 as reported in TSPLIB. Length: 429.983312.

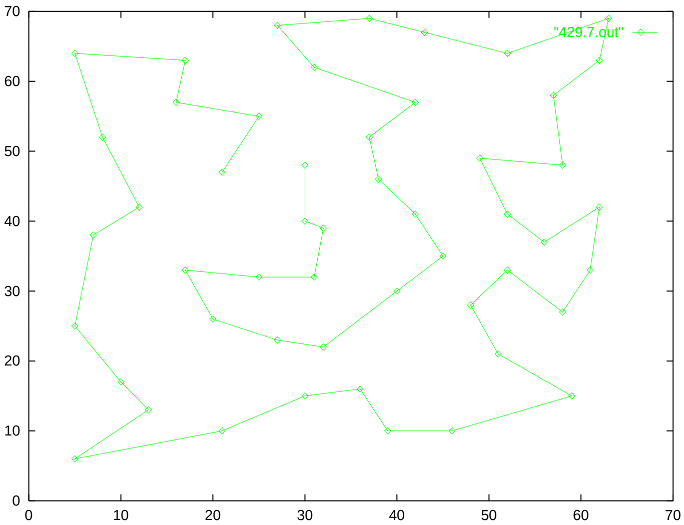

Figure 4: Tour found by present study. Length: 429.737129. Tour: 6 48 23 7 43 24 14 25 13 41 19 40 42 44 15 45 33 39 10 49 9 30 34 21 50 16 2 29 20 35 36 3 28 31 26 8 22 1 32 11 38 5 37 17 14 18 47 12 46 51 27

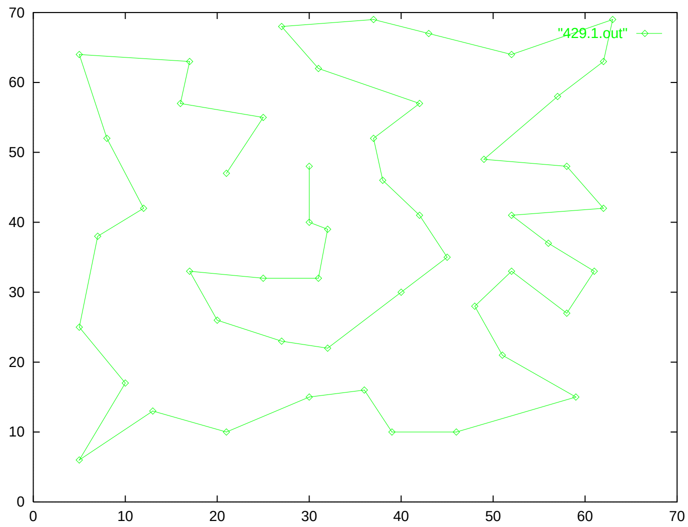

Figure 5: Tour found by present study. Length: 429.117939. Tour: 6 48 23 7 43 24 14 25 13 41 40 19 42 44 15 45 33 39 10 49 9 30 34 50 16 21 29 2 20 35 36 3 28 31 26 8 22 1 32 11 38 5 37 17 4 18 47 12 46 51 27

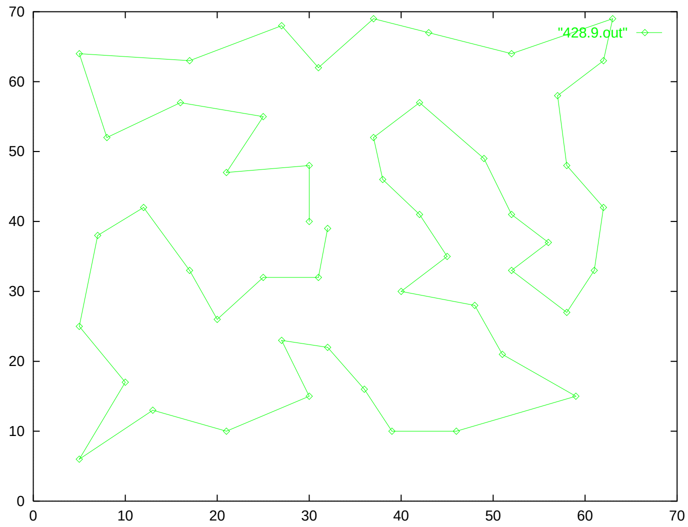

Figure 6: Tour found by present study. Length: 429.981647. Tour: 46 12 47 4 18 14 25 13 41 40 19 42 44 17 37 15 45 33 39 10 49 5 38 11 32 1 22 2 16 50 9 30 34 21 29 20 35 36 3 28 31 8 26 7 43 24 23 48 6 27 51

GA to evolve the parameters of their ant-based algorithms, in this case routing algorithms for communications networks. Unfortunately, their algorithms were not tested on any common benchmark problem and it is therefore hard to comment on their results.

Although the method presented in this paper looks promising, a lot more work needs to be done. The method has to be tested on many more TSP instances, including significantly larger ones. However, the approach is computation-intensive, as it requires running ACS on each problem 40 times per generation. To overcome this issue, it would be useful to understand how the best parameters scale with problem size and tour length. Once this is known, the parameters obtained for small problems can be generalized to larger problems. This, of course, assumes that there exist well-defined scaling laws.

Preliminary results on including a form of *division of labor* (Robinson, 1992) into the ACO algorithm, with groups of ants characterized by different parameters within the same colony, are also promising. The existence of three classes (or castes) of ants was assumed, each class being characterized by a different value of the exploitation parameter q_0. Five parameters were evolved: three values of q_0 (one per caste) and the fractions of ants in each caste (two parameters since the three fractions add up to make 1). All other parameters remained fixed in the course of the evolutionary search (parameters from Dorigo and Gambardella (1997a,b)). Extremely good solutions were found in the first generations—so that the GA did not make much progress after that. A set of parameters was obtained that resulted in the optimal solution of Oliver30 being found after only 70 ant-cycles (7 cycles with 10 ants). More tests are required to confirm this result.

Finally, although we have used a GA to evolve the parameters of an ACO algorithm, the very same approach can be used to evolve the parameters of other optimization algorithms (Lin *et al.*, 1993).

5. Acknowledgments

H. B. was supported by the National Science Foundation's REU Program during this work. E. B. is the Interval Research Fellow at the Santa Fe Institute. We thank Marco Dorigo for providing us with some of his papers prior to publication.

References

Beckers, R., Deneubourg, J.-L., Goss, S. and Pasteels, J. M. (1990). Collective decision making through food recruitment. *Ins. Soc.* **37**, 258–267.

Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., Aron, S. and Camazine, S. (1997). Self-organization in social insects. *Trends in Ecol. Evol.* **12**, 188–193.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1998). *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY: Oxford University Press.

Colorni, A., Dorigo, M. and Maniezzo, V. (1991). Distributed optimization by ant colonies. *Proc. First Europ. Conf. on Artificial Life*, (Varela, F. and Bourgine, P., eds), pp. 134–142. Cambridge, MA: MIT Press.

Colorni, A., Dorigo, M., Maniezzo, V. and Trubian, M. (1993). Ant system for job-shop scheduling. *Belg. J. Oper. Res., Stat. and Comput. Sci.* **34**, 39–53.

Deneubourg, J.-L. and Goss, S. (1989). Collective patterns and decision making. *Ethol. Ecol. and Evol.* **1**, 295–311.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Trans. Syst. Man Cybern. B* **26**, 29–41.

Dorigo, M. and Gambardella, L.M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Trans. Evol. Comp.* **1**, 53–66.

Dorigo, M. and Gambardella, L.M. (1997). Ant Colonies for the Traveling Salesman Problem. *BioSystems* **43**, 73–81.

Eilon S., Watson-Gandy, C. D. T. and Christofides, N. (1969). Distribution management: mathematical modeling and practical analysis. *Op. Res. Quart.* **20**, 37–53.

Gambardella, L. M., Taillard, E. D. and Dorigo, M. (1998). Ant colonies for the QAP *J. Operational Res. Soc.* (in press).

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley.

Lawler, E. L., Lenstra, J. K., Rinnooy-Kan, A. H. G. and Shmoys, D. B. (eds) (1985). *The travelling salesman problem.* New York, NY: Wiley.

Lin, F.-T., C.-Y. Kao, and C.-C. Hsu (1993). Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems. *IEEE Trans. Syst. Man Cybern.* **23**, 1752–1767.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell Syst. Journal* **44**, 2245–2269.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.* **21**, 498–516.

Maniezzo, V. and Colorni, A. (1998). The ant system applied to the quadratic assignment problem. *IEEE Trans. Knowledge and Data Engineer.* (in press).

Riolo, R.L. (1992). Survival of the Fittest Bits. *Scientific American* **267**, 114–116.

Robinson, G. E. (1992). Regulation of division of labor in insect societies. *Annual Rev. Entomol.* **37**, 637–665.

Schoonderwoerd, R., Holland, O., Bruten, J. and Rothkrantz, L. (1997). Ant-based load balancing in telecommunications networks. *Adapt. Behav.* **5**, 169–207.

White, T., Pagurek, B. and Oppacher, F. (1998). Connection management using adaptive mobile agents. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '98)* (Arabnia, H. R., ed), pp. 802–809. CSREA Press.

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