Inspiration for optimization from social insect behaviour

E. Bonabeau*†, M. Dorigo‡ & G. Theraulaz§

* *Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, New Mexico 87501, USA*
† *Eurobios, Tour Ernst & Young, 92037 La Défense Cédex, France*
‡ *IRIDIA, Université Libre de Bruxelles, CP 194/6, Avenue Franklin Roosevelt 50, 1050 Brussels, Belgium*
§ *Laboratoire d'Ethologie et Cognition Animale, CNRS-FRE 2041, Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cédex, France*

Research in social insect behaviour has provided computer scientists with powerful methods for designing distributed control and optimization algorithms. These techniques are being applied successfully to a variety of scientific and engineering problems. In addition to achieving good performance on a wide spectrum of 'static' problems, such techniques tend to exhibit a high degree of flexibility and robustness in a dynamic environment.

Ethologists use modelling to understand animal behaviour. Recent research in social insect behaviour suggests that models based on self-organization can help explain how complex colony-level behaviour emerges out of interactions among individual insects[1–3]. Although the goal of a modeller is generally to understand the living, a model can in principle be explored beyond the biologically plausible. Although biology does not necessarily benefit from such an exploration, computer scientists and engineers have been able to transform models of social insect collective behaviour into useful optimization and control algorithms. This new line of research concerns the transformation of knowledge about how social insects collectively solve problems into artificial problem-solving techniques—producing a form of Artificial Intelligence, or Swarm Intelligence[3], in which the underlying model of intelligence is the collective intelligence of a social insect colony. Some of the techniques of Swarm Intelligence are now maturing. Optimization and control algorithms inspired by models of co-operative food retrieval in ants have been unexpectedly successful and have become known in recent years as Ant Colony Optimization (ACO)[4,5] and Ant Colony Routing (ACR)[6–8]. Real-world implementations are emerging. Other techniques, inspired by co-operative brood sorting by ants[9,10] or task allocation in social wasps[3], are still in a preliminary, proof-of-concept stage, with no systematic benchmarking of their performance. ACO and ACR will be described in more detail here.

Ant Colony Optimization

The ability of ants to optimize is best exemplified by the experiment described in Box 1. In this experiment[1], the ants can select the shortest path to a food source because they lay and follow pheromone (chemical) trails. But the ants' success in collectively selecting the shortest path is only statistical: the colony may occasionally get 'stuck' on a longer path if by chance the longer path is the first one to be marked. In using the 'trail laying–trail following' metaphor for optimization purposes, computer scientists have found it essential to improve the convergence properties of their algorithms by artificially increasing the rate of pheromone evaporation beyond biological plausibility.

The classic Travelling Salesman Problem (TSP), which consists of finding the shortest tour between n cities visiting each once only and ending at the starting point, illustrates how the use of artificial pheromones can be applied to hard optimization problems[4,5]. Each artificial ant visits the n cities to construct a tour. After completing its tour it lays 'artificial pheromone' on the links it has used in an amount that is proportional to the quality of the tour, so that links which belong to good solutions end up with more pheromone than the other links. To construct a tour, each ant tends to select a city connected to its current city by a link that has more pheromone than the others: this selection process amplifies previously reinforced links and leads to the emergence of a good solution. Pheromone evaporates at a fixed rate after all ants have constructed their tours. Evaporation prevents mediocre links from being amplified by accident. The mathematical description of the simplest ACO algorithm described in Box 2 is a proof-of-concept, which performs poorly on large TSP instances. More sophisticated versions include a local search that seeks to improve existing solutions by evaluating the effects of small changes (such as swapping two cities), a more complex tradeoff between exploration and exploitation, increased reinforcement of the best tour, and other refinements. Such improvements make the algorithm much more competitive, in terms of both computing time and quality of solutions[5,40]: comparisons of an improved ACO algorithm with other efficient, state-of-the-art implementations of general-purpose approaches for the TSP

Box 1

How ants find the shortest path

Ant colonies can collectively perform tasks and make decisions that appear to require a high degree of co-ordination among the workers: building a nest, feeding the brood, foraging for food, and so on. In the example presented here, the ants collectively discover the shortest path to a food source. In experiments with the ant *Linepithaema humile*, a food source is separated from the nest by a bridge with two branches[1,2]. The longer branch is r times longer than the shorter branch (Fig. 2a).

The shorter branch is selected by the colony in most experiments if r is sufficiently large ($r = 2$ in Fig. 2b). This is because the ants lay and follow pheromone trails: individual ants lay a chemical substance, a pheromone, which attracts other ants. The first ants returning to the nest from the food source are those who take the shorter path twice (from the nest to the source and back). At choice points 1 and 2, nestmates are recruited toward the shorter branch, which is the first to be marked with pheromone. If, however, the shorter branch is presented to the colony after the longer branch, the shorter path will not be selected because the longer branch is already marked with pheromone (Fig. 2b).

This problem can be overcome in an artificial system, by introducing pheromone decay: when the pheromone evaporates sufficiently quickly, it is more difficult to maintain a stable pheromone trail on a longer path. The shorter branch can then be selected even if presented after the longer branch. This property is particularly desirable in optimization, where one seeks to avoid convergence toward mediocre solutions. In *Linepithaema humile*, although pheromone concentrations do decay, the lifetime of trail pheromones is too large to allow such flexibility.

Box 2

Ant Colony Optimization and the Travelling Salesman Problem

The Travelling Salesman Problem (TSP) consists of finding the shortest tour between n cities visiting each once only and ending at the starting point. Let d_{ij} be the distance between cities i and j and τ_{ij} the amount of pheromone on the edge that connects i and j. τ_{ij} is initially set to a small value τ_0, the same for all edges (i,j). The algorithm consists of a series of iterations. One iteration of the simplest ACO algorithm applied to the TSP can be summarized as follows: (1) a set of m artificial ants are initially located at randomly selected cities; (2) each ant, denoted by k, constructs a complete tour, visiting each city exactly once, always maintaining a list J_k of cities that remain to be visited; (3) an ant located at city i hops to a city j, selected among the cities that have not yet been visited, according to probability $p_{ij}^k = ([\tau_{ij}]^\alpha \cdot [d_{ij}]^{-\beta})/(\Sigma_{l \in J_k}[\tau_{il}]^\alpha \cdot [d_{il}]^{-\beta})$, where α and β are two positive parameters which govern the respective influences of pheromone and distance; (4) when every ant has completed a tour, pheromone trails are updated: $\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}$, where ρ is the evaporation rate and $\Delta\tau_{ij}$ is the amount of reinforcement received by edge (i,j). $\Delta\tau_{ij}$ is proportional to the quality of the solutions in which (i,j) was used by one ant or more. More precisely, if L_k is the length of the tour T_k constructed by ant k, then $\Delta\tau_{ij} = \Sigma_{k=1}^{m}\Delta\tau_{ij}^k$, with $\Delta\tau_{ij}^k = Q/L_k$ if $(i,j) \in T_k$ and $\Delta\tau_{ij}^k = 0$ otherwise, where Q is a positive parameter. This reinforcement procedure reflects the idea that pheromone density should be lower on a longer path because a longer trail is more difficult to maintain.

Steps (1) to (4) are repeated either a predefined number of times or until a satisfactory solution has been found. The algorithm works by reinforcing portions of solutions that belong to good solutions and by applying a dissipation mechanism, pheromone evaporation, which ensures that the system does not converge early toward a poor solution. When $\alpha = 0$, the algorithm implements a probabilistic greedy search, whereby the next city is selected solely on the basis of its distance from the current city. When $\beta = 0$, only the pheromone is used to guide the search, which would reflect the way the ants do it. However, the explicit use of distance as a criterion for path selection appears to improve the algorithm's performance[4,5]. In all other optimization applications also, an improvement in the algorithm's performance is observed when a local measure of greed, similar to the inverse of distance for the TSP, is included into the local selection of portions of solution by the agents. Typical parameter values are: $m = n$, $\alpha = 1$, $\beta = 5$, $\rho = 0.5$, $\tau_0 = 10^{-6}$.

(genetic algorithm[27], iterated local search and iterated Lin-Kernighan, see http://wwwkeck.caam.rice.edu/concorde.html) show that the ACO algorithm is competitive, always providing among the best results.

In addition to finding very good solutions to 'static' problems, the algorithm also maintains a pool of alternative portions of solutions, which can be especially useful when the problem is dynamically changing: the algorithm can channel new searches toward this pool. This is because the pheromone update equation ensures that every link has at least a small amount of pheromone: a weakly used link can be quickly reinforced and replace a missing or failing link. Although there is no formal framework for quantifying the performance of optimization algorithms in dynamic environments, the flexibility and robustness of an algorithm are important factors when it comes to real-world implementations.

The TSP is a natural application of ACO, but many other combinatorial optimization problems can be solved with ACO. ACO is currently the best available heuristic for the sequential ordering problem and for real-world instances of the quadratic assignment problem, and is among the most competitive approaches for the vehicle and network routing problems, as well as for a number of other problems (see Table 1). Regarding real-world applications, an ACO implementation for multi-stage factory scheduling is under study at Unilever (D. Gregg *et al.*, personal communication), and gasoline trucks are now being routed with the help of an ACO algorithm in Italian Switzerland (L. M. Gambardella, personal communication).

ACO, however, does not always work well. For example, it does not perform as well as other heuristics on any problem's instances that have been uniformly randomly generated[3]. The reason is that ACO tends to reinforce portions of solutions that belong to many good solutions: the more good solutions a given portion belongs to, the more virtual pheromone it receives. If a large number of portions of solutions are equally likely to be part of good solutions, ACO cannot differentiate them and therefore performs poorly. However, many real-world problems do possess enough of the requisite 'structure' to allow this approach to be very efficient. Other optimization techniques, based on exotic neural networks[11] or evolutionary algorithms[12], rely on the same 'building block' principle, that is, they attempt to find the building blocks of good solutions by maintaining a search statistics. In ACO, the search statistics is summarized into pheromone concentrations, which, after a sufficient number of iterations, indicate the likelihood that particular portions of solution, or building blocks, belong to one of the best solutions. ACO not only identifies building blocks but also identifies correlations between building blocks: for example, two or more portions of solution may be desirable only if used together. This property is crucial, as most instances of most problems are not readily 'linearly' decomposable into building blocks.

Ant Colony Routing in communications networks

ACO has an additional feature that makes it particularly appealing: it is explicitly formulated in terms of computational agents. Although it may be, in principle, possible to focus only on the core optimizing mechanism (reinforcement of portions of solutions and global dissipation), the agent-based formulation is a useful aid for designing problem-solving systems. The example of routing in communications networks illustrates this point.

Routing is the control mechanism that directs every message in a communications network from its source node to its destination node through a sequence of intermediate nodes or switching stations. Each switching station has a routing table that tells messages or portions of messages called packets where to go, given their destinations. Because of the highly dynamic nature of communications networks, due to the time-varying stochastic changes in user traffic patterns as well as to unpredictable failures of network components, portions of a network may become congested and new routes have to be discovered dynamically. In Ant Colony Routing (ACR), first described by Schoonderwoerd *et al.*[6], ant-like agents reinforce routing-table entries depending on their experience and performance in the network[6–8]. For example, if

Table 1 Some applications of Ant Colony Organization and Ant Colony Routing

Problem name	Year	Main references	Algorithm name
Travelling salesman	1991	4	AS
	1997	5	ACS & ACS-3-opt
	1997	28	MMAS
Network routing	1997	6	ABC
	1998	7	Co-operative Asymmetric Forward
	1998	8	AntNet
	1999	29	ABC-backward
Graph colouring	1997	30	ANTCOL
Shortest common supersequence	1999	31	AS-SCS
Quadratic assignment	1999	32	HAS-QAP
	1999	33	MMAS-QAP
	1999	34	AS-QAP
Machine scheduling	1999	35	ACS-SMTTP
Vehicle routing	1999	36	AS-VRP
	1999	37	MACS-VRPTW
Multiple knapsack	1999	38	AS-MKP
Frequency assignment	2000	39	ANTS-FAP
Sequential ordering	1997	40	HAS-SOP

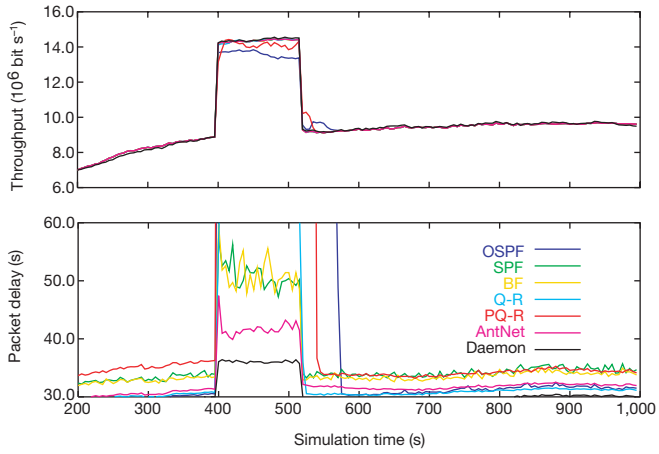

Figure 1 Typical result of a comparison of AntNet, an Ant Colony Routing algorithm, with other widespread routing algorithms for packet-switched networks (see ref. 41 for an overview of communications networks). OSPF is the Open Shortest Path First algorithm, the current official Internet routing algorithm[3,8]. BF is the asynchronous distributed Bellman-Ford algorithm with a dynamic cost metric[3,8,41]. SPF is the Shortest Path First algorithm with a dynamic cost metric[3,8]. Q-R is the Q-Routing algorithm[3,8]. PQ-R is the Predictive Q-Routing algorithm[3,8], an extension of Q-Routing. Daemon is an approximation of an ideal algorithm and provides a bound to the best performance achievable[3,8]. The network, the underlying architecture of which is that of NSFnet, is highly loaded with stochastic time-varying traffic. At time 400 a sudden increase in traffic, lasting 120 s, takes the network to saturation conditions. The upper graph compares throughput (the larger the better), while the lower graph packet delays averaged over a 5-s time window (the smaller the better). AntNet provides the same throughput as the best competing algorithms maintaining a much lower average packet delay. After ref. 8, courtesy of the AI Access Foundation and Morgan Kaufmann.

an agent has been delayed a long time because it went through a highly congested portion of the network, it will weakly reinforce routing-table entries that send messages to that portion of the network; an agent that enjoyed fluid traffic conditions will apply a stronger reinforcement. A dissipation or evaporation mechanism is also applied regularly to the routing-table entries to refresh the system and prevent obsolete solutions from being maintained.

When tested on simulated networks and realistic traffic conditions, ACR appears to outperform routing algorithms which are in widespread use, especially, but not only, in strongly variable traffic conditions[8] (Fig. 1). Maintaining a pool of alternate routes is the way the system copes with changing conditions, including node failure, and this technique makes it flexible and robust. Although some of the reported results are quite good, intensive real-world testing would be useful. Recent news articles indicate that British Telecom and MCI-Worldcom have been and still are using algorithms based on the ant colony metaphor[13].

Other applications inspired by social insects

Ant foraging is not the only social insect behaviour that has inspired computer scientists and roboticists. Other examples include division of labour, brood sorting and co-operative transport.

Division of labour. In most social insect species, individual workers tend to be specialized in certain tasks for a varying portion of their lifetime[14]. But the behavioural repertoire of workers can be stretched back and forth in response to perturbations: if needed, a forager can become a nurse, or a nurse a guard, and so forth. Such a combination of specialization and plasticity in task allocation is appealing for multi-agent optimization and control. A response threshold model of division of labour has been used to solve dynamic task scheduling problems[3,15]. In a threshold model[14], workers with low response thresholds respond to lower levels of stimuli than do workers with high response thresholds. Task performance reduces the intensity of stimuli. If workers with low thresholds perform their normal tasks the task-associated stimuli never reach the thresholds of the high-threshold workers. But if, for any reason, the intensity of task-associated stimuli increases, high-threshold workers engage in task performance. For example, to allocate trucks coming out of an assembly line to paint booths in a truck factory, each paint booth is considered to be an insect-like agent that is specialized in one colour; but if needed, the paint booth can change its colour (although this is costly). Thus, the swarm intelligent system minimizes task changeovers and can cope with glitches[3].

Brood sorting. In the ant *Leptothorax unifasciatus*, workers sort the brood[16]. Eggs and microlarvae are compactly clustered at the centre of the brood area; the largest larvae are located at the periphery of the brood cluster; when pupae and prepupae are present, they are located between peripheral large larvae and the more central larvae of medium size. Deneubourg *et al.*[17] have proposed a model of this phenomenon, recognizing that many of their model's assumptions had to be tested. In the model, an ant picks up and drops an item according to the number of similar surrounding items. For example, if an ant carries a large larva, it will drop the larva with high probability in a region populated by large larvae, or if an unladen ant finds a large larva surrounded by eggs, it will pick up the larva with high probability; in all other situations the ant will neither drop nor pick up any item. Although the model still needs validation, computer scientists have found it useful for data sorting. Lumer and Faieta[9] and Kuntz *et al.*[10] have applied it to the following problem. Given a data set of points in an n-dimensional space and a metric δ which measures the distance between pairs of data points, project the points onto the plane so that if any two projected points in the plane are neighbours, their corresponding data items are

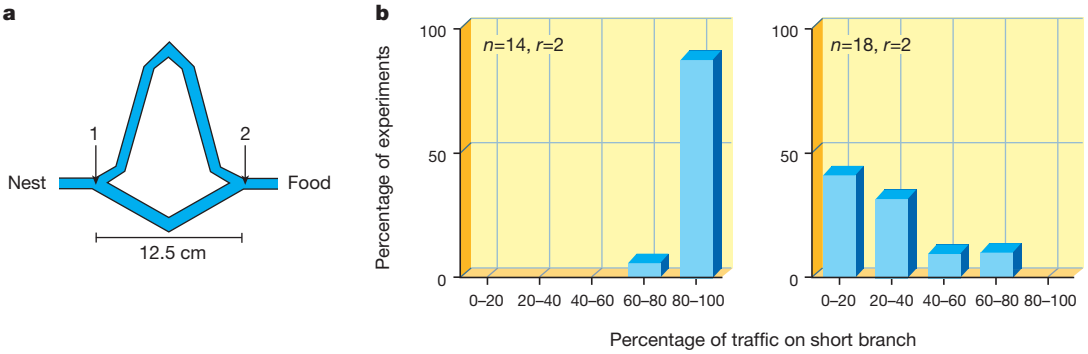

Figure 2 The double-bridge experiment. **a**, Experimental set-up. **b**, Distribution of the percentage of ants that selected the shorter branch over a set of experiments. The longer branch is r times longer than the short branch. The left graph (14 experiments) represents experiments in which both branches were presented simultaneously. The right graph (18 experiments) represents experiments in which the shorter branch was presented to the colony 30 minutes after the longer branch: the shorter branch is not selected and the colony keeps on exploiting the longer branch. Modified from ref. 2.

neighbours under the metric δ. The initial projection of data items onto the plane is random. The artificial ants then perform random walks on the plane and pick up or drop projected data items using rules from the model[17]. The results are qualitatively equivalent to those obtained by such classic techniques as spectral decomposition or stress minimization[10], but at a much lower computational cost.

Co-operative transport. Swarm Intelligence is also an inspiration for roboticists to design distributed control algorithms for groups, or swarms, of robots[18–22]. One example of a task that has been used as a benchmark for swarm robotics is co-operative transport, or more precisely co-operative box pushing[20,21]. Co-operative transport has been reported in several ant species[23–25]. When it is impossible for a single ant to retrieve a large prey item, nestmates are recruited. Then, for a few minutes, the ants change position and alignment around the prey item without making progress, until it can be moved toward the nest. This emergent co-ordination has been reproduced by Kube and Zhang[21] in a swarm of very simple robots whose task is to push a box toward a goal. Their work shows how a task that appears to require the co-ordinated efforts of several robots can be performed without explicit co-ordination or communication among the robots. Such work is promising in the perspective of miniaturization and low-cost robotics (see for example the Alice micro-robots at http://dmtwww.epfl.ch/isr/asl/projects/alice_pj.html).

Perspectives

The initial appeal of Swarm Intelligence to computer scientists was almost entirely due to their fascination with ants. The surprisingly good results obtained by several groups of researchers make it even more appealing for optimization and control applications, especially in view of the ability of swarm intelligent systems to cope with glitches and changing environments.

A number of problems, best exemplified by routing in telecommunications networks, are simply much better tackled with computational agents. The breakthrough brought by Swarm Intelligence is permitting the design of agent-based, distributed optimization and control techniques. We view Swarm Intelligence as a major new paradigm in optimization and control. Its closest relative would be the artificial neural network paradigm. Indeed, an ant colony is a 'connectionist' system, that is, one in which individual units are connected to each other according to a certain pattern. However, crucial differences from typical neural networks include: (1) the dynamic nature of the connectivity pattern, which can be exploited by examining how ants or other social insects solve problems; (2) the explicit or implicit mobility of the units, which is an essential feature of ACO and ACR; (3) feedback from the environment, which is used as a medium of co-ordination and communication; and (4) the use of pheromone evaporation in ACO, which dramatically facilitates the design of distributed optimization techniques.

We have no doubt that more practical applications of Swarm Intelligence will continue to emerge. In a world where a chip will soon be embedded into every object, from envelopes to trash cans to heads of lettuce, control algorithms will have to be invented to let all these 'dumb' pieces of silicon communicate[26]. Meanwhile, a better understanding of how and why methods based on social insects work and more systematic comparisons with other heuristics for optimization and control are needed. □

1. Deneubourg, J. -L. & Goss, S. Collective patterns and decision making. *Ethol. Ecol. Evol.* **1,** 295–311 (1989).
2. Goss, S., Aron, S., Deneubourg, J. -L. & Pasteels, J. M. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften* **76,** 579–581 (1989).
3. Bonabeau, E., Dorigo, M. & Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems* (Oxford Univ. Press, New York, 1999).
4. Dorigo, M., Maniezzo, V. & Colorni, A. The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. B* **6,** 29–41 (1996).
5. Dorigo, M. & Gambardella, L. M. Ant colonies for the traveling salesman problem. *BioSystems* **43,** 73–81 (1997).
6. Schoonderwoerd, R., Holland, O., Bruten, J. & Rothkrantz, L. Ant-based load balancing in telecommunications networks. *Adapt. Behav.* **5,** 169–207 (1997).
7. Heusse, M., Guérin, S., Snyers, D. & Kuntz, P. Adaptive agent-driven routing and load balancing in communication networks. *Adv. Compl. Syst.* **1,** 237–254 (1998).
8. Di Caro, G. & Dorigo, M. AntNet: Distributed stigmergetic control for communications networks. *J. Artif. Intell. Res.* **9,** 317–365 (1998).
9. Lumer, E. & Faieta, B. in *Proc. 3rd Intl Conf. Simulation of Adaptive Behavior: From Animals to Animats 3* (eds Cliff, D., Husbands, P., Meyer, J. -A. & Wilson, S. W.) 501–508 (MIT Press, Cambridge, MA, 1994).
10. Kuntz, P., Snyers, D. & Layzell, P. A stochastic heuristic for visualizing graph clusters in a bi-dimensional space prior to partitioning. *J. Heuristics* **5,** 327–351 (1999).
11. Chen, K. A simple learning algorithm for the traveling salesman problem. *Phys. Rev. E* **55,** 7809–7812 (1997).
12. Baluja, S. & Caruana, R. in *Proc. 12th Intl Conf. Machine Learning* (eds Prieditis, A. & Russell, S.) 38–46 (Morgan Kaufmann, Palo Alto, 1995).
13. Ward, M. There's an ant in my phone. *New Sci.* **2118,** 32–35 (1998).
14. Robinson, G. E. Regulation of division of labour in insect societies. *Annu. Rev. Entomol.* **37,** 637–665 (1992).
15. Sobkowski, A. *et al.* in *Bio-Computation and Emergent Computing* (eds Lundh, D., Olsson B. & Narayanan, A.) 36–45 (World Scientific, Singapore, 1997).
16. Franks, N. R. & Sendova-Franks, A. B. Brood sorting by ants: distributing the workload over the work surface. *Behav. Ecol. Sociobiol.* **30,** 109–123 (1992).
17. Deneubourg, J. -L. *et al.* in *Proc. 1st Conf. Simulation of Adaptive Behavior: From Animals to Animats* (eds Meyer, J. A. & Wilson, S. W.) 356–365 (MIT Press, Cambridge, MA, 1991).
18. Cao, Y. U., Fukunaga, A. S. & Kahng, A. B. Cooperative mobile robotics: antecedents and directions. *Autonomous Robots* **4,** 7–27 (1997).
19. Martinoli, A., Yamamoto, M. & Mondada, F. in: *Proc. 4th European Conf. Artificial Life* (eds Husbands, P. & Harvey, I.) (MIT Press, Cambridge, MA, 1997).
20. Nilsson, M. & Simsarian, K. T. in *Proc. 1995 IEEE/RSJ Intl Conf. Intelligent Robots and Systems* 556–561 (IEEE Computer Society Press, Los Alamitos, 1995).
21. Kube, C. R. & Zhang, H. Collective robotics: from social insects to robots. *Adaptive Behavior* **2,** 189–218 (1994).
22. Kube, C. R. & Zhang, H. Task modelling in collective robotics. *Autonomous Robots* **4,** 53–72 (1997).
23. Sudd, J. H. The transport of prey by ants. *Behaviour* **25,** 234–271 (1965).
24. Franks, N. R. Teams in social insects: group retrieval of prey by army ants (*Eciton burchelli*, Hymenoptera: Formicidae). *Behav. Ecol. Sociobiol.* **18,** 425–429 (1986).
25. Moffett, M. W. Cooperative food transport by an asiatic ant. *Nat. Geogr. Res.* **4,** 386–394 (1988).
26. Kelly, K. *New Rules for the New Economy* (Viking, New York, 1998).
27. Walters, T. in *Proc. PPSN V, Conference on Parallel Problem-Solving from Nature* (eds Eiben, A. E., Bäck, T., Schoenauer, M. & Schwefel, H.-S.) 813–822 (Springer, Berlin, 1998).
28. Stützle, T. & Hoos, H. in *Proc. IEEE Intl Conf. Evolutionary Computation* (eds Bäck, T., Michalewicz,Z. & Yao,X.), 309–314 (IEEE Computer Society Press, Los Alamitos, 1997).
29. Van der Put, R. & Rothkrantz, L. in *Simulation Practice and Theory* (in the press).
30. Costa, D. & Hertz, A. Ants can colour graphs. *J. Op. Res. Soc.* **48,** 295–305 (1997).
31. Michel, R. & Middendorf, M. in *Proc. PPSN V, Conference on Parallel Problem-Solving from Nature* (eds Eiben, A. E., Bäck, T., Schoenauer, M. & Schwefel, H.-S.) 692–701 (Springer, Berlin, 1998).
32. Gambardella, L. M., Taillard, E. D. & Dorigo, M. Ant colonies for the quadratic assignment problem. *J. Op. Res. Soc.* **50,** 167–176 (1999).
33. Stützle, T. & Hoos, H. in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization* (eds Voss, S., Martello, S., Osman, I. H. & Roucairol, C.) 313–329 (Kluwer Academic, Boston, 1999).
34. Maniezzo, V. & Colorni, A. The ant system applied to the quadratic assignment problem. *IEEE Trans. Knowledge Data Engin.* **11,** 769–778 (1999).
35. Bauer, A., Bullnheimer, B., Hartl, R. F. & Strauss, C. in *Proc. Congr. Evolutionary Computation (CEC'99)* 1445–1450 (IEEE Press, Piscataway, NJ, 1999).
36. Bullnheimer, B., Hartl, R. F. & Strauss, C. in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization* (eds Voss, S., Martello, S., Osman, I. H. & Roucairol, C.) 109–120 (Kluwer Academic, Boston, 1999).
37. Gambardella, L. M., Taillard, E. D. & Agazzi, G. in *New Ideas in Optimization* (eds Corne, D., Dorigo, M. & Glover, F.) 63–76 (McGraw-Hill, London, 1999).
38. Leguizamón, G. & Michalewicz, Z. in *Proc. Congr. Evolutionary Computation (CEC'99)* 1459–1464 (IEEE Press, Piscataway, NJ, 1999).
39. Maniezzo, V. & Carbonaro, A. An ANTS heuristic for the frequency assignment problem. *Future Generation Comput. Syst. J.* **16,** 927–935 (2000).
40. Gambardella, L. M. & Dorigo, M. Ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS J. Comput.* (in the press).
41. Bertsekas, D. & Gallager, R. *Data Networks* (Prentice Hall, Englewood Cliffs, 1992).

Acknowledgements

E. B. is supported by the Interval Research Fellowship at the Santa Fe Institute. E. B. and G.T. are supported in part by a grant from the GIS (Groupement d'Intérêt Scientifique) Sciences de la Cognition. G. T. is supported by a grant from the Conseil Régional Midi-Pyrénées. M. D. acknowledges support from the Belgian FNRS, of which he is a Research Associate.

Correspondence and requests for materials should be addressed to E.B. (e-mail: eric.bonabeau@eurobios.com). Information and data on comparisons with other algorithms, including source codes, can be obtained from M.D. (e-mail: mdorigo@ulb.ac.be).