Chapter 4

Ant colony optimization

Ants are among the most widespread living organisms on the planet, and they are native to all regions except Antarctica and certain islands (such as, for example, Greenland). Ants display a multitude of fascinating behaviors, and one of their most impressive acheivements is their amazing ability to cooperate in order to achieve goals far beyond the reach of an individual ant. An example is the dynamic bridge-building exhibited by some species of ants. When such ants encounter a gap (between two trees, say) that is too wide to be crossed by a single ant, they form a bridge using their own bodies. Another example is the collective transport of heavy objects, exhibited by many species of ants. In this case, groups of ants cooperate to move an object (i.e. a large insect) that is beyond the carrying capacity of a single ant. While perhaps seemingly simple, this is indeed a very complex behavior: First of all, the ant that discovers the object must realize that the object is too heavy to transport without the help of other ants. Next, in order to obtain the help of additional ants, the first ant must carry out a process of **recruitment**. It does so by secreting a substance from a poison gland, attracting nearby ants (up to a distance of around 2 m). Then, the transport must be coordinated regarding e.g. the number of participating ants, their relative positions around the object etc. Furthermore, there must be a procedure for overcoming deadlocks, in cases where the transported object becomes stuck. Studies of ants have shown that their capability for collective transport is indeed remarkable: As an example, groups of up to 100 ants, carrying objects weighing 5,000 times the weight of a single ant have been spotted [7].

The complex cooperative behaviors of ants have inspired researchers and engineers. For example, the dynamic bridge-building mentioned above has inspired research on rescue robots. In this application, the scenario is as follows: Freely moving robots are released in a disaster area to search for injured or unconscious victims of the disaster. If, during the search, a robot encounters an insurmountable obstacle, it may recruit other robots to form a bridge, or some

other suitable structure, to pass the obstacle, and then continue the search.

Ants are also capable of remarkably efficient foraging (food gathering). Watching ants engaged in this behavior, one may believe that the ants are equipped with very competent leaders and are capable of communicating over long distances. However, neither of these assertions would be true: Ants self-organize their foraging, without any leader, and they do so using mainly short-range communication. In fact, the essential features of ant foraging can be captured in a rather simple set of equations that underlie the ant colony optimization (ACO) algorithms that will be studied in this chapter. However, we shall begin with a brief discussion of the biological background for ACO.

4.1 Biological background

In general, cooperation requires communication. Even though direct communication (e.g. by sound) occurs in ants, their foremost means of communication is indirect. Ants are able to secrete substances known as **pheromones**, and thus to modify the environment in a way that can be perceived by other ants. This form of indirect communication through modification of the local environment is known as **stigmergy** and it is exhibited not only by ants but by many other species (e.g. termites) as well.

During foraging, ants traverse very large distances, in some cases up to 100 m, before returning to the nest carrying food. This is an amazing feat, considering that the size of a typical ant is of order 1 cm. How is it achieved? The answer is that, upon returning to the nest, an ant will deposit a trail of pheromone. Other ants that happen to encounter the pheromone scent, are then likely to follow it, thus reinforcing the trail. The trail-making pheromones deposited by ants are volatile hydrocarbons. Hence, the trail must be continually reinforced in order to remain detectable. Therefore, when the supply of food (from a given source) drops, the trail will no longer be reinforced, and will eventually disappear altogether. The local communication mediated by pheromones is thus a crucial component in ant behavior.[1]

The stigmergic communication in ants has been studied in controlled experiments by Deneubourg *et al.* [3]. In their experiments, which involved ants of the species *L. humile* (Argentine ants), the ants' nest was connected to a food source by a bridge, shown schematically in Fig. 4.1. As can be seen from the figure, an ant leaving the nest is given a choice of two different paths, at two

[1]In fact, pheromones have many other uses as well, both in ants and other species. A fascinating example is that of slave-making ants: about 50 of the 12,000 or so species of ants, are slave-makers, i.e. they invade the nests of other ants and capture their workers. The exact invasion procedure varies between species, but some species are capable of using pheromones in a kind of chemical warfare: Releasing their pheromones, they cause members of the other species to fight among themselves, allowing the invading species to go about its business.

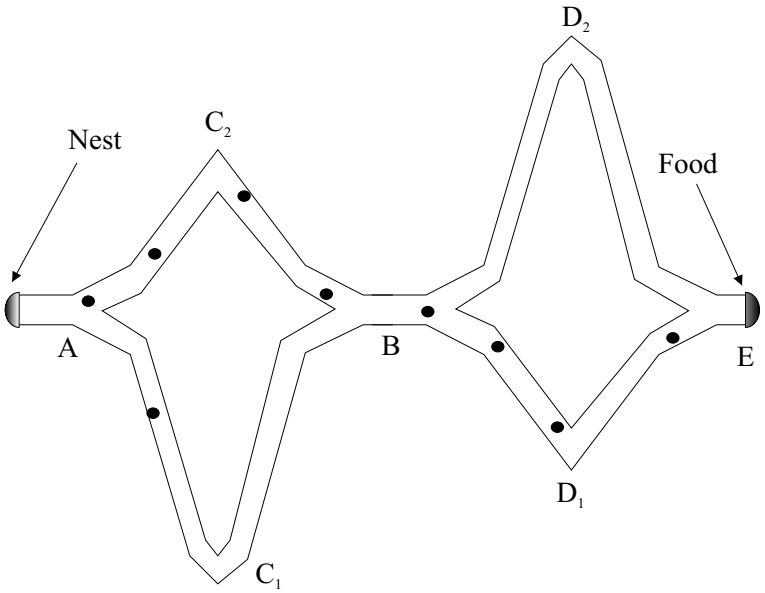

Figure 4.1: *A schematic view of the experimental setup used by Deneubourg* et al. *[3].*

points (A and B) along the way towards the food source. The second half of the path might seem unnecessary, but it was included to remove any bias: Thus, at the first decision point, the short path segment is encountered if a left turn is made whereas, at the second decision point, an ant must make a right turn in order to find the short path segment. Furthermore, the shape of the branches (at the decision points A and B) was carefully chosen to avoid giving the ants any guidance based on the appearance of the path: Both branches involved an initial 30° turn, regardless of which branch was chosen.

Thus, the first ants released into this environment made a random choice of direction at both decision points: some ants would take a long path (A-C_1-B-D_2-E), some a short path (A-C_2-B-D_1-E), and some a path of intermediate length. However, those ants that happened to take the shortest path, in both directions would, of course, return to the nest before the other ants (assuming roughly equal speed of movement). Thus, at this point, ants leaving the nest would detect a (weak) pheromone scent upon reaching A, and would therefore display a preference for the shorter path. Upon their return, they would then deposit pheromone along this path, further reinforcing the trail. In the experiment, it was found that, a few minutes after the discovery of the food source, the shortest path was clearly preferred.

In addition to the experiment, Deneubourg *et al.* also introduced a simple numerical model for the dynamics of trail formation, which we will now consider briefly. Since the model deals with artificial ants, there is no need to use the elaborate path considered in the experiment with real ants. Thus, a simplified path with a single decision point in each direction, shown in Fig. 4.2,

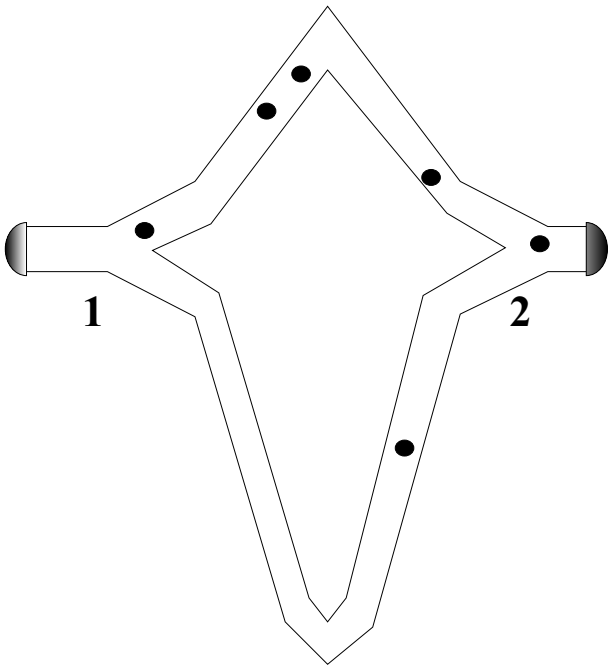

Figure 4.2: *A simplified setup, used in numerical experiments of ant navigation.*

can be used. The numerical model accounts for the fact that *L. humile* deposit pheromones both on the outbound and the inbound part of the motion. Thus, upon arriving at decision point 1, while moving towards to food source, an ant chooses the short path with a probability p_1^S, empirically modelled as

$$p_1^S = \frac{(C + S_1)^m}{(C + S_1)^m + (C + L_1)^m}, \tag{4.1}$$

where S_1 and L_1 denote the amount of pheromone along the short and long paths, respectively, at the decision point, and C and m are constants. Furthermore, before proceeding, the ant deposits a unit of pheromone on the chosen branch. Since the ants must choose one of the two paths, the probability of p_1^L of selecting the long path equals $1 - p_1^S$. Similarly, while inbound (i.e. moving towards the nest), arriving at decision point 2 the ant chooses the short path with probability

$$p_2^S = \frac{(C + S_2)^m}{(C + S_2)^m + (C + L_2)^m}, \tag{4.2}$$

and then again deposits a unit of pheromone on the chosen branch.

In the experiments with real ants, traversing a short branch (e.g. A-C_2-B in Fig. 4.1) took around 20 s, whereas an ant choosing a long branch would need $20r$ s, where r is the length ratio of the two branches. It was found that a good fit to experimental data (for various values of r in the range $[1, 2]$) could be found for $C = 20$ and $m = 2$.

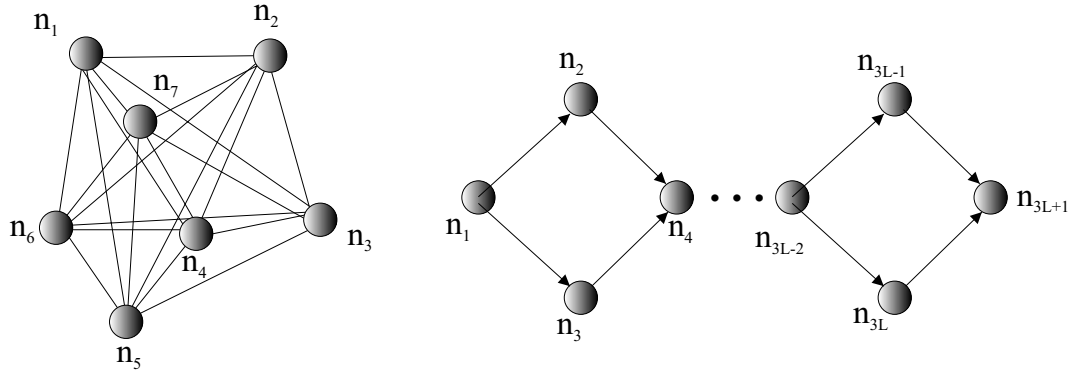

Figure 4.3: *Construction graphs for ACO. Left panel: A typical construction graph for TSP. Right panel: A chain construction graph.*

4.2 Ant algorithms

Given the biological background of ACO, i.e. the foraging behavior of ants it is, perhaps, not surprising that ACO algorithms operate by searching for paths (representing the solution to the problem at hand) in a graph. Thus, applying ACO is commonly a bit more demanding than using a GA or PSO since, in order to solve a problem using ACO, it is necessary to formulate it as the problem of finding the optimal (e.g. shortest) path in a given graph, known as a **construction graph**. Such a graph, here denoted G, can be considered as a pair (N, E), where N are the nodes (vertices) of the graph, and E are the edges, i.e. the connections between the nodes. In ACO, artificial ants are released onto the graph G, and move according to probabilistic rules that will be described in detail below. As the ants generate their paths over G, they deposit pheromone on the edges of the graph. Thus, each edge e_{ij}, connecting node j to node i is associated with a pheromone level τ_{ij}. The exact nature of the construction graph G varies from problem to problem.

The left panel of Fig. 4.3 shows a typical construction graph for one of the most common applications of ACO, namely the **travelling salesman problem** (TSP). The aim, in TSP, is to find the shortest path that visits each city once (and only once) and, in the final step, returns to the city of origin. For TSP the number of possible paths, which equals $(n-1)!/2$ where n is the number of nodes, grows very rapidly as the size of the problem is increased. In standard TSP, the nodes $n_i \in N$, represent the cities, and the edges e_{ij} the (straight-line) paths between them. Thus, for this problem, the construction graph has a very straightforward implementation: It is simply equivalent to the **physical graph** in which the actual movement takes place. In fact, because of its straightforward interpretation, TSP is an ideal problem for describing ACO, and we will use it in the detailed description of ACO algorithms below.

However, in many problems, the construction graph is an abstract object,

and functions merely as a vehicle for finding solutions to the problem at hand. The right panel of Fig. 4.3 shows a **chain construction graph**, which generates a binary string that can then be used e.g. as the input to a Boolean function. A chain construction graph that generates a binary string of length L consists of $3L+1$ nodes and $4L$ edges. Starting from the initial node n_1, either an up-move or a down-move can be made, the decision being based on the pheromone levels on the edges e_{12} and e_{13}. If an up-move is made (to n_2), the first bit of the string is set to 1, whereas it is set to 0 if a down-move (to n_3) is made. The next move, to the center node n_4, is deterministic, and merely serves as a preparation for the next bit-generating step, which is carried out in the same way as the first move. Thus, upon reaching the end of the chain, the artificial ant will have generated a string consisting of L bits.

Since the introduction of ACO by Dorigo [4, 5], several different versions have been developed. Here, we will consider two versions, namely Ant System (AS) and Min-Max Ant System (MMAS), and we will use the special case of the TSP to describe the algorithms.

4.2.1 Ant System

Ant system was the first ACO algorithm introduced [5]. In order to solve the TSP over n nodes, a set of M artificial ants are generated and released onto the construction graph. The formation of a candidate solution to the problem involves traversing a path that will take the ant to each city (node) once, and then return to the origin. Thus, each ant starts with an empty candidate solution $S = \emptyset$ and, for each move, an element representing the index of the current node is added to S. In order to ascertain that each node is visited only once, each ant maintains a memory in the form of a **tabu list** $L_T(S)$, containing the indices of the nodes already visited. Thus, initially, the node of origin is added to L_T. As mentioned above, each edge e_{ij} of the graph is associated with a pheromone level τ_{ij}. In any given step, an ant chooses its move probabilistically, based on a factor depending on the pheromone level as well as the distances between the current node and the potential target nodes. More specifically, the probability of the ant taking a step from node j to node i is given by

$$p(e_{ij}|S) = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{e_{kj}\notin L_T(S)} \tau_{kj}^{\alpha}\eta_{kj}^{\beta}}, \tag{4.3}$$

where, in TSP, $\eta_{ij} = 1/d_{ij}$, where d_{ij} is the (Euclidean) distance between node j and node i. α and β are constants that determine the relative importance (from the point of view of the artificial ants) of the pheromone trails and the problem-specific information η_{ij}. In each step of the movement, the current node is added to the tabu list L_T. When all nodes but one have been visited, only the as yet unvisited node is absent from the tabu list, and the move to that

node therefore occurs with probability 1, as obtained from Eq. (4.3). Next, the tour is completed by a return to the city of origin (which is obtained as the first element of L_T).

Eq. (4.3) specifies, albeit probabilistically, the movement of the ants, given the pheromone levels τ_{ij}. However, the equation says nothing regarding the variation of pheromone levels. In order to complete the description of the algorithm, this crucial element must also be described. In AS, an **iteration** consists of the evaluation of all M ants. At the end of each iteration, all ants contribute to the pheromone levels on the edges of the construction graph. Let D_m denote the length of the tour generated by ant m. The pheromone level on edge e_{ij} is then increased by ant m according to

$$\Delta\tau_{ij}^m = \begin{cases} \frac{1}{D_m} & \text{if ant } m \text{ traversed the edge } e_{ij}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

The total increase in the pheromone level on edge e_{ij} is thus given by

$$\Delta\tau_{ij} = \sum_{m=1}^{M} \Delta\tau_{ij}^m. \tag{4.5}$$

Obviously, with Eq. (4.5), pheromone levels will increase indefinitely. However, taking a cue from the properties of real pheromone trails, AS introduces pheromone evaporation, and the complete update rule thus takes the form

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}, \tag{4.6}$$

where $\rho \in]0, 1]$ is the **evaporation rate**. Note that Eq. (4.6) is applied to *all* edges in the construction graph. Hence, for edges e_{ij} that are not traversed by any ant, the pheromone update consists only of evaporation. AS is summarized in Algorithm 4.1.

The free parameters in AS are the constants α, β and ρ, as well as M (the number of ants). The optimal values of these parameters will, of course, vary between problems. However, experimental studies have shown that, over a large range of problems, the values $\alpha = 1$, $\beta \in [2, 5]$, $\rho = 0.5$, and $n = M$ (i.e. the number of nodes in the construction graph) yield adequate performance A remaining issue is the initialization of the pheromone trails. In AS, it is common to initialize the pheromone levels as $\tau_{ij} = \tau_0 = n/D^{[nn]}$, where $D^{[nn]}$ is the length of the tour obtained by, at each step, moving to the nearest (as yet unvisited) node.

4.2.2 Max-Min Ant System

MMAS, introduced by Stützle and Hoos [9], is a version of AS that attempts to exploit good candidate solutions more strongly than AS. While MMAS shares

1. Initialize pheromone levels:
   $\tau_{ij} = \tau_0,\ \forall (i,j) \in [1,n]$
2. For each ant m, select a random starting node, and add it to the (initially empty) tabu list L_T. Next, build the tour S. In each step of the tour, select the move from node j to node i with proability $p(e_{ij}|S)$, given by:
   $p(e_{ij}|S) = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{e_{kj}\notin L_T(S)}\tau_{kj}^{\alpha}\eta_{kj}^{\beta}}$.
   In the final step, return to the node of origin, i.e. the first element in L_T. Finally, compute and store the length D_m of the tour.
3. Update the pheromone levels:
   3.1. For each ant m, determine $\Delta\tau_{ij}^m$ as:
   $$\Delta\tau_{ij}^m = \begin{cases} \frac{1}{D_m} & \text{if ant } m \text{ traversed the edge } e_{ij}, \\ 0 & \text{otherwise,} \end{cases}$$
   3.2. Sum the $\Delta\tau_{ij}^m$ to generate $\Delta\tau_{ij}$:
   $\Delta\tau_{ij} = \sum_{m=1}^{M}\Delta\tau_{ij}^m$.
   3.3. Modify τ_{ij}:
   $\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \Delta\tau_{ij}$.
4. Repeat steps 2 and 3 until a satisfactory solution has been found.

Algorithm 4.1: *Ant system (AS), applied to the case of TSP. See the main text for a complete description of the algorithm.*

many features with AS, there are also important differences between the two algorithms. First of all, in MMAS, only the ant generating the best solution is allowed to deposit pheromone. The *best solution* can either be defined as the *best in the current iteration* or *the best-so-far*. Thus, in MMAS, letting D_b denote the length of the best tour,

$$\Delta\tau_{ij} = \Delta\tau_{ij}^{[b]}, \tag{4.7}$$

where $\tau_{ij}^{[b]} = 1/D_b$ if the best ant traversed the edge e_{ij} in its tour, and 0 otherwise. Note that, as in AS, pheromone levels are updated for *all* edges, according to Eq. (4.6), even though only n edges receive a non-zero $\Delta\tau_{ij}$. Thus, the edges that are not in the tour of the best ant are subject only to pheromone evaporation.

Second, since the strong exploitation of good solutions may lead to stagnation, where all ants essentially follow the same trail, in MMAS limits are introduced on the pheromone levels. Thus, after the pheromone update according to Eq. (4.6), the levels are modified such that if $\tau_{ij} > \tau_{\max}$ then $\tau_{ij} \leftarrow \tau_{\max}$ and, likewise, if $\tau_{ij} < \tau_{\min}$, then $\tau_{ij} \leftarrow \tau_{\min}$.

A third difference between MMAS and AS concerns the pheromone initialization. In MMAS, pheromone levels are initialized to the maximum value $\tau_{\max}$, i.e. initally,

$$\tau_{ij} = \tau_{\max} \ \forall (i,j) \in [1,n], \tag{4.8}$$

The choice of $\tau_{\max}$ is based on estimates of the theoretical upper limit of the pheromone levels under the update rule given in Eqs. (4.6) and (4.7). In fact, it can easily be shown (see Appendix C) that an upper bound of the pheromone deposited on any edge e_{ij} is given by $1/(\rho D^*)$, where D^* is the length of the optimal tour. Obviously, during the optimization procedure D^* is not yet known. Therefore, $\tau_{\max}$ is set to $1/(\rho D_b)$, the length of the current best tour. Whenever a new best tour is found, $\tau_{\max}$ is updated using the new value of D_b. $\tau_{\min}$ is often chosen through experimentation.

Furthermore, for MMAS, it is possible to prove (see Appendix C) that, as the number of iterations tends to infinity, the probability of finding the optimal solution tends to 1.

Note that, despite the pheromone limits, it is not uncommon that the MMAS algorithm gets stuck on a suboptimal solution. Thus, a useful approach is to restart the algorithm, using the most recent available estimate of D^* (i.e. the current D_b), whenever no improvements are detected over a number of iterations. It should also be noted that, at least for TSP with large (> 200) values of n, better results are obtained by alternating the definition of the best solution. Thus, the *best in the current iteration* is used when updating the pheromones, for a number of iterations, after which a switch is made to the *best so far* etc.

4.3 Applications

In view of their biological background, it is not surprising that ACO algorithms have been applied to a variety of problems involving routing. The TSP, described above is perhaps the foremost example, but ACO algorithms have also been used for solving other routing problems involving, for example, telecommunication networks [8]. Here, however, we shall consider two other applications, starting with machine scheduling.

4.3.1 Single-maching scheduling

In the general problem of **job shop scheduling**, n jobs, consisting of a finite sequence of operations, are to be processed on m machines, in the shortest possible time, while fulfilling constraints regarding the order of precedence between the different operations in the jobs. This is an example of a **scheduling problem**, for which ACO algorithms have been developed. Here we shall consider another scheduling problem, namely the **single-machine total weighted**

tardiness problem (SMTWTP). In this problem, the aim is to schedule the order of execution of n jobs, assigned to a single machine. Let t_j denote the processing time of job j, and d_j its due date (i.e. the deadline). The tardiness T_j of a job j is defined as

$$T_j = \max(0, c_j - d_j), \tag{4.9}$$

where c_j is the actual completion time of the job. Thus, jobs that are completed on time receive a tardiness of 0 whereas, for jobs that are not completed on time, the tardiness equals the delay. The aim of the **single-machine total tardiness problem** (SMTTP) is to minimize the total tardiness T_{tot}, given by

$$T_{\text{tot}} = \sum_{j=1}^{n} T_j, \tag{4.10}$$

whereas, in the weighted problem (SMTWTP), the goal is to minimize the weighted total tardiness, defined as

$$T_{\text{tot,w}} = \sum_{j=1}^{n} w_j T_j. \tag{4.11}$$

In this problem, any permutation of the n jobs is a feasible solution. Thus, in order to generate a solution using an ACO algorithms, the ants must build a sequence of jobs, such that each job is executed exactly once. As in TSP, the ants base their choice on a combination of two factors: the pheromone level τ_{ij} involved in placing job i immediately after job j, and a problem-specific factor η_{ij} (see Eq. (4.3)). In TSP, the latter factor involved the inverse of the distance between the nodes j and i. In SMT(W)TP, by contrast, the problem-specific factor involves a measure of time, instead of distance. Several different measures have been used in the literature, e.g.

$$\eta_{ij} = \frac{1}{t_j}, \tag{4.12}$$

a measure referred to as the **shortest processing time** (SPT), which is effecient in cases where most jobs are late. Other measures involve the due date; the simplest such measure

$$\eta_{ij} = \frac{1}{d_j}, \tag{4.13}$$

is referred to as **earliest due date** (EDD). The two measures above are obviously simple to obtain, but may not be the most efficient, since they do not take into account the growing sequence of scheduled jobs. An alternative is to use the **modified due date** (MDD), defined as

$$\eta_{ij} = \frac{1}{\max(T^{[s]} + t_j, d_j)}, \tag{4.14}$$

where $T^{[s]}$ denotes the total processing time of the jobs already scheduled. In order to use this measure, the factors η_{ij} must, of course, be updated after each addition to the job sequence.

Using any of the measures η_{ij} above, an ACO algorithm (e.g. AS, as described above) can easily be set up to solve the SMT(W)TP. However, it should be noted that, for this particular problem, it is common to use another version of ACO, known as **ant colony system** (ACS) which, however, is beyond the scope of this text. In addition, the candidate solutions to the SMT(W)TP obtained by the artificial ants are usually enhanced by **local search**. This procedure, which can be applied either to all candidate solutions in the current iteration or (to reduce the amount of computation) only to the best candidate solution of the current iteration, involves searching for neighbouring candidate solutions, using e.g. **interchange**, in which all $n(n-1)/2$ interchanges of pairs of jobs are considered. Thus, for example, if the job sequence is $(1, 6, 4, 7, 3, 5, 2)$ (in a simple case involving only seven jobs), an interchange of the second and fifth jobs in the sequence results in $(1, 3, 4, 7, 6, 5, 2)$. An alternative procedure is **insertion**, in which the job at position i_1 in the sequence is extracted and reinserted in position i_2. Thus, again considering the sequence $(1, 6, 4, 7, 3, 5, 2)$, extraction of the second job, followed by reinsertion in the fifth position, results in the sequence $(1, 4, 7, 3, 6, 5, 2)$. Obviously, the two processes of interchange and insertion can also be combined.

In cases with small n (up to around 50), classical integer programming techniques, such as the **branch-and-bound** method can be counted on to solve the SMT(W)TP. However, for larger n, ACO-based methods become useful. It should be noted that the level of difficulty in solving randomly generated instances of SMT(W)TP is strongly variable: For some instances of the problem the best solution can be found quite easily whereas, for others, finding the best solution can be very challenging. Thus, in order to investigate performance, a given method must be tested against multiple instances of the problem. A procedure for generating instances of SMTTP is given in [1]. Here, the processing times t_j for the n jobs are integers chosen from the uniform distribution in the range $[1, n]$, and the due dates d_j are also integers taken from the uniform distribution in the range D given by

$$D = \left[\sum_{j=1}^{n} t_j \left(1 - c_1 - \frac{c_2}{2} \right), \sum_{j=1}^{n} t_j \left(1 - c_1 + \frac{c_2}{2} \right) \right], \tag{4.15}$$

where c_1 is the **tardiness factor** and c_2 is the **relative range of due dates**. The complexity of the problem varies with c_1 and c_2. In [1], these parameters were both taken from the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. For each of the $5 \times 5 = 25$ possible combinations, five instances of the problem were generated for $n = 50$ and $n = 100$. Using ACO the optimal (or, rather, best known) solution was found in 124 out of 125 cases for $n = 50$, and in all 125 cases for $n = 100$. In [2], similar

results were reported for the SMTWTP. These two studies show that, for the SMT(W)TP, ACO algorithms generally outperform other stochastic methods.

4.3.2 Cooperative transportation using autonomous robots

As mentioned in the introduction to this chapter, many species of ants are capable of cooperative transportation of objects, a process that has inspired robotics researchers to investigate cooperative behaviors in autonomous robotics. Ants participating in cooperative transport appear to coordinate their motions based on stigmergy, i.e. indirect communication, something that also makes the procedure suitable for robotic applications by avoiding the problem of direct communication between the participating robots. In fact, however, little is known about the detailed processes involved in ant movement coordination, and therefore cooperative transport may turn out to be an instance where the connection between biology and robotics goes both ways, i.e. where robotics may also inspire further biological research

Cooperative transportation using autonomous robots has been studied by, among others, Kube and Zhang [6] using both simulations and actual robots. In those experiments, a group of identical autonomous (wheeled) robots were given the task of locating a box near the centre of an arena, and then, cooperatively, push the box to the edge of the arena. Of course, the box was too heavy to be pushed by a single robot: In order for the box to move, a minimum of two robots, pushing in roughly the same direction, was required. The problem is illustrated schematically in Fig. 4.4, and it was solved in the framework of **behavior-based robotics**, in which the control system of a robot is built in a bottom-up fashion from a set of elementary behaviors. Starting with simulation experiments, Kube and Zhang delevoped a robotic control system consisting of five elementary behaviors, namely *Find* (for locating the box), *Slow* (for slowing down, so as to avoid rear-end collisions with another robot), *Follow* (for following another robot), *Avoid* (for avoiding collisions), and *Goal* (for directing a robot towards the box). In addition, a structure for **behavior selection**, i.e. for timely activation of the various behaviors was generated. A full description of this mechanism will not be given here. Suffice it to say that the experiments were successful, resulting in reliable cooperative transport of the box (in a more or less random direction, depending on the exact distribution of robots around the box). Building upon the lessons learned in the simulation experiments, Kube and Zhang were also able to transfer their results successfully to real robots, Furthermore, the problem of deadlocks was considered. Each robot was equipped with a counter, measuring the time t elapsed since the latest move. In case the box got stuck, such that the robots kept pushing without generating any movement, t would, of course, eventually exceed a threshold T, at which point the robot would change the direction

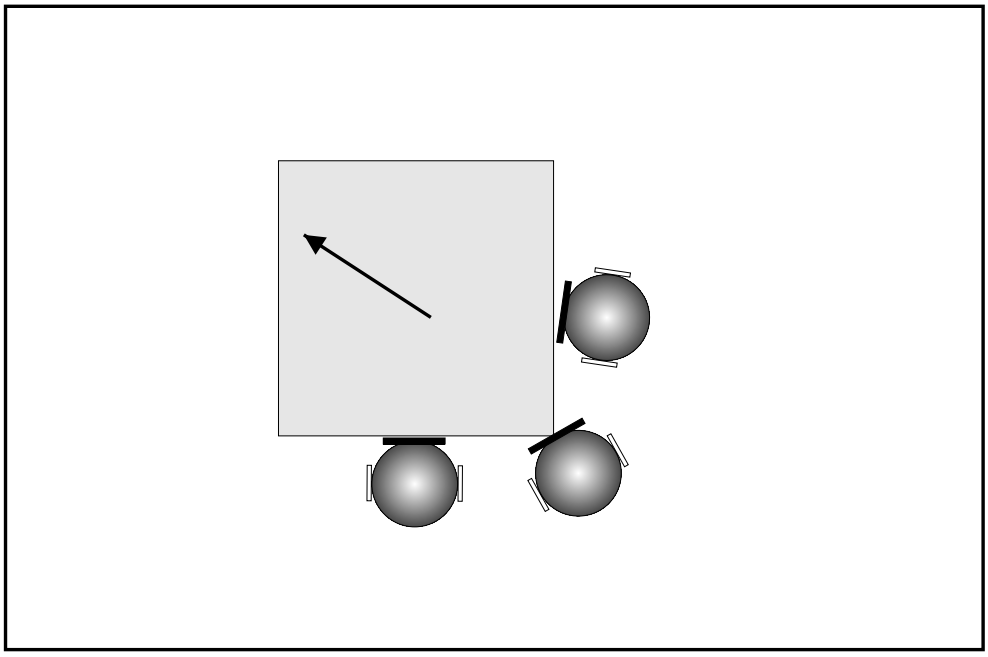

Figure 4.4: *Schematic illustration of a box-pushing task, in which three wheeled robots are attempting to move a large, rectangular box towards the edge of the arena. The arrow indicates the approximate instantaneous direction of motion.*

of the applied force. If this also failed, repositioning was tried, i.e. the robot in question would move to a different location around the perimeter of the box, and continue pushing.

Even though this simple box-pushing task may not appear to be very relevant at first glance, the mere fact that successful cooperative transport could be generated is important, as it paves the way for more complex, real-world applications, for example in rescue missions. In such cases, a swarm of robots would be released in a disaster area (resulting from e.g. an earthquake or a fire) to search for disaster victims. A robot discovering an injured person would then recruit other robots to the scene, and the group of robots would cooperatively move the person to safety.

In addition to the experiments described by Kube and Zhang, other studies of ant-inspired cooperative robot behaviors have been carried out within the framework of the **Swarm-bot project** [10] This project involved the development of a fairly simple, cost-effective robot, that could be mass-produced for use in multi-robot applications. Several tasks, such as dynamic bridge-building in order to pass a large obstacle, were considered within the framework of the Swarm-bot project.

Appendix C: Convergence proof

Pheromone limits in MMAS

Proposition For the Min-Max Ant System (MMAS) algorithm, the maximum pheromone level $\tau_{\max}$ on any edge e_{ij} is bounded (asymptotically) by f^*/ρ, where f^* is the fitness of the optimal solution (i.e. $1/D^*$ in the case of TSP).

Proof Let τ_0 denote the initial amount of pheromone on the edges of the construction graph. In any given iteration, the maximum amount of pheromone that can be added to an edge e_{ij} equals f^*. Thus, after the first iteration, the maximum pheromone level equals $(1-\rho)\tau_0 + f^*$, and after the second generation it equals $(1-\rho)^2\tau_0 + (1-\rho)f^* + f^*$. In general, after the K^{th} iteration, the maximum amount of pheromone equals

$$\tau_{\max}(K) = (1-\rho)^K\tau_0 + \sum_{k=1}^{K}(1-\rho)^{k-1}f^*. \tag{C1}$$

Since $\rho \in]0,1]$, in the limit $K \to \infty$, we obtain

$$\tau_{\max} = f^*/\rho, \tag{C2}$$

using the well-known result

$$\sum_{k=1}^{\infty}(1-\rho)^{k-1} = 1/\rho. \tag{C3}$$

Convergence proof

Theorem Let p(K) be the probability that the optimal solution (for TSP) is encountered (using MMAS) at least once in the first K generations. Then

$$\lim_{K\to\infty} p(K) = 1. \tag{C4}$$

Proof In MMAS, pheromone levels are bounded from below by $\tau_{\min}$, and from above by $\tau_{\max}$. In fact, the proposition above shows that, due to pheromone evaporation, an explicit upper bound is not even needed. Consider now the construction of a tour by a given ant. We can determine a lower bound for the probability [2] of traversing any edge e_{ij}, by setting the pheromone level

[2]For simplicity the problem-specific factor η_{ij} will be dropped (i.e. set to 1) in the proof. However, as long as $0 < \eta_{\min} \leq \eta_{ij} \leq \eta_{\max} < \infty$, $\forall(i,j) \in [1,n]$, the proof still holds even with η_{ij} included.

on that edge to $\tau_{\min}$, and the pheromone level on all other edges to $\tau_{\max}$. The probability $p_{\min}$ of traversing e_{ij} is thus bounded from below by p_w, given by

$$p_{\min} \geq p_w = \frac{\tau_{\min}^{\alpha}}{(n-1)\tau_{\max}^{\alpha} + \tau_{\min}^{\alpha}}. \tag{C5}$$

Thus, any tour (including an optimal tour S^*) will be generated with a probability p_S fulfilling

$$p_S \geq p_{\min}^{n}. \tag{C6}$$

Thus, the probability of at least *one* ant finding an optimal solution is bounded from below by

$$p(K) = 1 - (1 - p_S)^K. \tag{C7}$$

Thus, evidently,

$$\lim_{K \to \infty} p(K) = 1. \tag{C8}$$

Bibliography

[1] Bauer, A., Bullnheimer, B., Hartl, R.F., and Strauss, C. *Minimizing total tardiness on a single machine using ant colony optimization*, Central European Journal for Operations Research and Economics, **8**, pp. 125-141, 2000

[2] den Besten, M.L., Stützle, T., Dorigo, M. *Ant colony optimization for the total weighted tardiness problem* In: Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN-VI), Lecture Notes in Computer Science, **1917**, pp. 611-620, 2000

[3] Deneubourg, J.-L., Aron, S., Goss, S., and Pasteels, J.-L. *The self-organizing exploratory pattern of the Argentine ant*, J. Insect Behavior **3**, pp. 159-168, 1990

[4] Dorigo, M. *Optimization, learning and natural algorithms* [in Italian], PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, 1992

[5] Dorigo, M., Maniezzo, V., and Colorni, A. *Ant System: Optimization by a colony of cooperating agents*, IEEE Trans. Systems, Man, and Cybernetics, Part B, **26**, pp. 29-41, 1996

[6] Kube, C.R., Zhang, H. *Collective Robotics: From Social Insects to Robots*, Adaptive Behavior **2**, pp. 189-218, 1994

[7] Moffett, M.W. *Cooperative food transport by an Asiatic ant*, National Geog. Res. **4**, pp. 386-394, 1988

[8] Schoonderwoerd, R.O., Holland, O., Bruten, J., and Rothkrantz, L. *Ant-based load balancing in telecommunications networks*, Adaptive Behavior, **5**, pp. 169-207, 1996

[9] Stützle, T. and Hoos, H.H. Max-Min Ant System, Future Generation Computer Systems, **16**, pp. 889-914, 2000

[10] `http://www.swarm-bots.org`