an $(1+1)$-EA is not the resulting runtime bound on its own. Significant are the underlying techniques that can be generalized to analyze EAs in different and more general situations. In [STW04], EAs for sorting are presented. The authors study different objective functions and mutation operators for the respective problem. The resulting runtime bounds for sorting n elements with an EA range from $\Theta(n^2 \cdot \log(n))$ to an in n exponentially increasing lower bound, depending on the exact choice of the objective function measuring the "sortedness" of unsorted sequences and the choice of the mutation operator.

Further examples for combinatorial optimization problems, which can be solved with an EA within certain, formally proved time bounds, are the single source shortest path (SSSP) problem ([STW04]), the maximum matching (MM) problem ([GW03]) and the minimum spanning tree (MST) problem ([NW08]).

In [Jäg03], one can find an attempt to analyze $(1+1)$-EAs for continuous objective functions $\mathbb{R}^D \to \mathbb{R}$. The author presents a version of the $(1+1)$-EA with an adapted mutation operator, where, roughly speaking, the variance of the mutation is altered depending on the success of the algorithm. For functions that behave like the Euclidean distance to some point $o \in \mathbb{R}^D$, i. e., for functions f satisfying

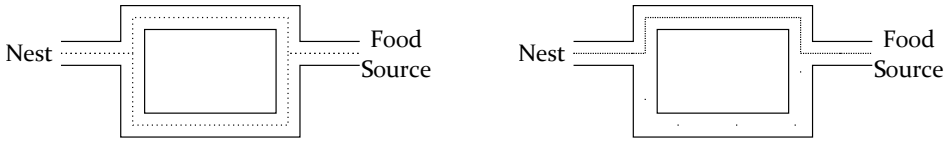
$$|x - o| < |y - o| \Rightarrow f(x) < f(y),$$

the author proves that the time for halving the distance between the individual and the global minimum o is linear in D.

2.7.2 Ant Algorithms

Ant algorithms are inspired by the famous double bridge experiment of Goss et al. ([GADP89]). A colony of Argentine ants was set into an artificial environment consisting of their nest and a food source which could be arrived via two different ways with different lengths. The situation is shown in Figure 2.10. In the beginning, the ants randomly decide for one of the two paths since the colony has no information obtained yet. While moving along the path, each ant emits pheromones, marking the path it has chosen. The ants that have decided for the shorter path arrive at the food source first and most likely take the same way back to the nest because until the other group also arrives at the food source, the pheromones are only on the shorter path.

When the ants that decided for the longer path finally arrive at the food source, they sense a higher concentration of pheromones on the shorter path than on the longer path because the shorter path is already traversed and marked twice. As more time passes and the ants go several rounds to the food source and back, the intensity difference between the pheromone values on the two ways further increases and the shorter path becomes more and more attractive. In the end, only very few ants use the longer way while the vast majority prefers the shorter path.



(a) At the beginning: Ants moving along one of the two paths.

(b) After some time: Most ants moving along the shorter path.

Figure 2.10: The double bridge experiment.

This capability of ants to find the shortest way to the food source has been modeled in order to create an optimization algorithm for the TSP ([DG97]). The resulting ant algorithm works as follows. Every edge $\{i, j\}$ of the input graph G is assigned a *pheromone value* $\tau_{i,j}$, which is initialized with some positive value. The pheromone values represent the memory of the colony. High pheromone values indicate a high probability for the ants to select the respective edge again. Additionally, every edge $\{i, j\}$ has a certain *visibility* $\eta_{i,j}$, also called *heuristic information*, a value that determines how attractive an edge is for the ants, without taking pheromones into account. A typical choice is $\eta_{i,j} = 1/d_{i,j}$, where $d_{i,j}$ is the length of edge $\{i, j\}$.

Then, until some termination criterion holds, the following simulation of ants' behavior is repeated: The colony of N ants is placed on the nodes of the graph, e. g., all the ants can be placed on the same node or a different node could be selected randomly for every ant. The variable π_k describes the partial tour that ant k has already traveled. At the beginning of the tour, π_k contains only the starting node of ant k. As long as the tour is not complete,

every ant k selects its next node randomly. If the current node of ant k is node i, then it moves to node j with probability $p_{i,j}^{(k)}$, defined as

$$p_{i,j}^{(k)} := \begin{cases} \frac{(\tau_{i,j})^{\alpha} \cdot (\eta_{i,j})^{\beta}}{\sum_{s \notin \pi_k} (\tau_{i,s})^{\alpha} \cdot (\eta_{i,s})^{\beta}} & \text{if } j \notin \pi_k \\ 0 & \text{otherwise,} \end{cases} \tag{2.12}$$

where α and β are positive constants controlling the influence of τ and η. Typical choices are $\alpha = 1$ and $\beta = 2$ ([DG97]). When the ants have com-

Algorithm 3: Ant algorithm

input : complete graph $G = (V, E)$ with n nodes and edge weights $d_{i,j}$ for $\{i, j\} \in E$
output: ordering π of the nodes of G

```
for {i, j} ∈ E do
    Initialize τ_{i,j} ;            // initialize pheromone values
    Calculate η_{i,j} ;        // calculate heuristic information
repeat
    for k = 1 → N do
        Initialize π_k ;                // initialize ants' tours
    for ℓ = 2 → n do
        for k = 1 → N do
            Set i := π_k(ℓ − 1);
            Choose j ∈ V with probability p_{i,j}^{(k)} as defined in Equation
            (2.12);
            Set π_k(ℓ) := j;
    Update τ according to Equation (2.13);
until Termination criterion holds;
return Best π_k;
```

pleted their tour, the pheromone values are updates via the equation

$$\tau_{i,j} := \rho \cdot \tau_{i,j} + \Delta\tau_{i,j}, \tag{2.13}$$

where $\rho \in (0, 1)$ is the pheromone decay parameter, describing how much of the pheromone vanishes during one tour of the ants. The value $\Delta\tau_{i,j}$ de-

scribes, by how much the pheromone value of edge $\{i, j\}$ is increased by ants moving over it. One example for a concrete choice of $\Delta\tau_{i,j}$ is ([DG97])

$$\Delta\tau_{i,j} = \sum_{k=1}^{N} \Delta\tau_{i,j}^{(k)}$$

with

$$\Delta\tau_{i,j}^{(k)} = \begin{cases} 1/L^{(k)} & \text{if edge } \{i, j\} \text{ is traversed by ant } k, \\ 0 & \text{otherwise,} \end{cases}$$

where $L^{(k)}$ is the length of ant k's tour. When the termination criterion holds, the algorithm is stopped and the best tour visited by any ant is returned.

An algorithmic overview over the ant algorithm can be found in Algorithm 3. For an overview over variants of the ant algorithm for binary problems, including some theoretical runtime results, see [NSW09].