**Swarm Intelligence - Ant Colony Optimization**

Ant Colony Optimization (ACO) is a numerical optimization technique inspired by the shortest path finding capabilities of ants and has been successfully applied to a wide range of optimization problems. After briefly introducing the biological background, this chapter will describe the ACO meta-heuristic and how it can be applied to solve NP-hard optimization problems such as the “Traveling Salesman Problem” (TSP) and “Quadratic Assignment” (QA).

**Biological Inspiration**

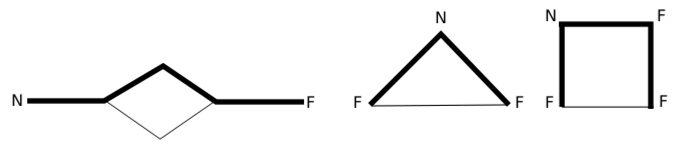
In the 80ies of last century, researchers have begun to systematically studying the trail laying characteristics of a variety of ant species. Ants are quickly able to collectively find what comes close to a shortest path between their nest and a food source (Goss, 1989), organize multiple such paths in “Minimal Spanning Trees”, and are highly efficient in covering large areas of terrain during raids (Deneubourg, 1989).



*The symmetric structure of a wasp’s nest, the structure of a honeycomb (and distribution of honey, larvae and pollen), and transport of food items are all mediated by indirect communication known as “Stigmergy”.*

It turns out that ants accomplish these feats exclusively via indirect communication through the environment by means of depositing “pheromones”. Pheromones are a chemical substance that can be secreted and detected by ants. Pheromones are still not fully understand in terms of what - and how much - information is encoded. Indirect communication with the environment is indeed ubiquitous in social insects and known as “stigmery” (Grassé, 1959). The word is derived from latin “stigma”, meaning “mark” and the greek word “ergon” that can be translated to “work”, and refers to any modification of the environment that stimulates either the same or other workers to some response. Examples include depositing grains of sand that will eventually lead to larger pillars in a termite mound, the geometry of a wasp nest or a honeycomb, but also simple task allocation such as brood sorting, removal of dead corpses, or storage of food.

Using a series of simple experiments subjecting ants to various bridge networks (Goss, 1989), researchers have been able to validate hypotheses on how ants could possible find a shortest path. In a nutshell, the intensity of a pheromone trail serves as recruitment leading to more ants choosing this path (positive feedback). Due to random fluctuations, one path will end up with slightly more pheromones than the other, leading to further reinforcement, and ultimately leading to almost all ants choosing the same path, even if  the arms of the double bridge are of equal length. This property will lead to ants constructing a “Minimal Spanning Tree” (MST) between all their destinations, including food sources and nest.



*Double bridge and other artificial foraging networks consisting of a nest (N) and one ore more food sources (F). Dominant routes are drawn with bold lines. Ants will self-organize the trail network into a minimum spanning tree (Goss, 1989).*

In case one arm is shorter, ants will arrive faster at the junction when they return to the nest. These additional pheromones reinforce this path, quickly leading to  this path becoming the dominant one. The researchers carefully counted the number of ants that have been passing over each branch of the bridge, here called *m1* and *m2*. The probability to select the first branch is then given by

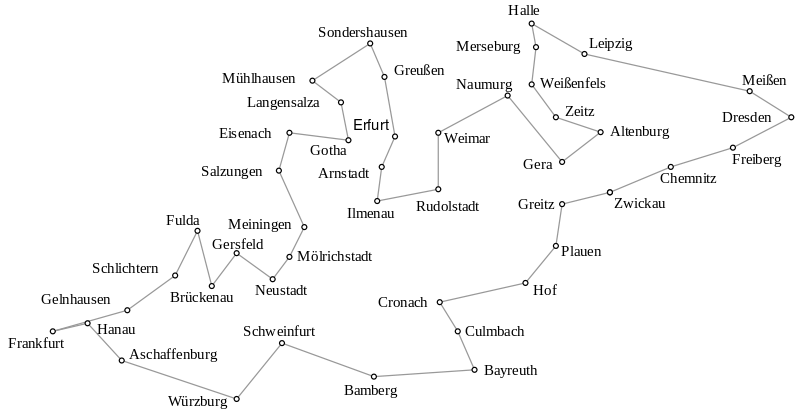
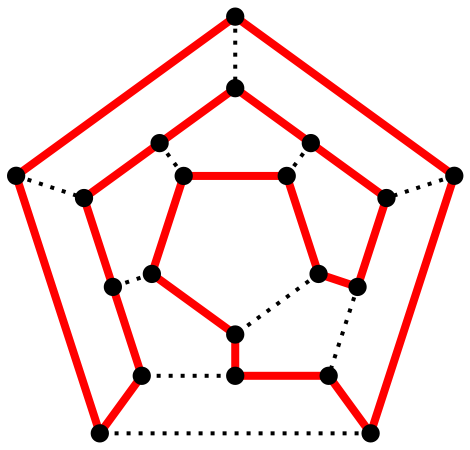
(1) https://lh3.googleusercontent.com/mGouDSYTIt8RhhtXA25k3s4jf5UKQE1VIsIpDNgjENhfo-VBhpLxJxbZchLtdjLVnLuuM2iKWg9DLl9Brex4nvZscul0AXFVZw_aSbRALrHwLNAEQRPGp81zGrKytnWmKRs-4Rcf

with the best fit for experimental data for k=20 and h=2 when simulated.

**Computationally hard Path Planning problems**

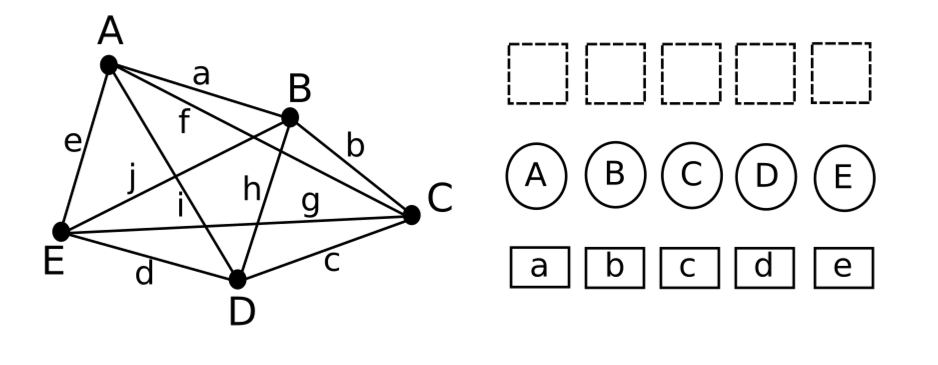
The biological model proposed by researchers around Jean-Louis Deneubourg has quickly inspired using software agents with similar behavior for solving tough optimization problems such as the “Traveling Salesman Problem” (TSP). Given a set of *N* cities with known locations, the goal is to find the shortest route that connects them all and returns at the origin. More technically, this problem consists of finding a minimum-length Hamiltonian cycle on a fully connected graph.

Assuming each city to be associated with character A, B, C, D and so forth, a feasible path *P* is given by a string of length *M*. By calculating the distance between each consecutive element, we can now associate a total length, also known as “cost”, with each path. The challenge is now to find the permutation of *P* that has the lowest cost. With *M!*, that is *M\*(M-1)\*(M-2)\*...*, and so forth, we can see that the number of paths quickly gets out of hand. Of course some paths, for example those that have cities that are very far apart directly following each other in the solution, are unlikely to be shortest. Similarly, it is possible to get a solution to the problem by simply following a greedy approach, picking the closest city every time. Yet, it can be shown that the TSP problem is NP-hard, and there exist problems for which optimal solutions are still not known.

**

*Left: A Hamiltonian cycle on a graph © Christoph Sommer CC-SA 3.0. The path visits every node only once and returns to its origin. Right: A Hamiltonian cycle over a network of cities, also known as the “Traveling Salesman Problem” © Thore Husfeldt CC-SA 3.0.*

The idea brought forward by one of the first computational adoptions of ant behavior (Dorigo, 1996) is simple. Let a virtual ant follow a semi-greedy heuristic that randomly picks among the closest cities to its current location using a similar probability than proposed by Goss et al. and leave artificial pheromones behind on each edge that it crossed. Every time a connection between two cities has been used, its pheromone level gets increased, and thereby the likelihood that another ant will choose it. In order to reflect the notion of distance and time, ants deposit pheromones that are inversely proportional to the path lengths. Best results are achieved if this is been done after a complete tour has been constructed, allowing to weigh each edge not only by its length, but also by its importance to the overall quality of the solution. Initially yielding rather poor results, Ant System quickly evolved into a powerful metaheuristic (see below) with a number of refinements, yielding highly competitive solutions for TSP and other, related optimization problems in NP.



*Left: A 5-city TSP problem. Finding an optimal tour requires finding an optimal assignment of all cities with places in the tour. Alternatively, the problem can be encoded as a sequence of edges, which represent both previous and next city.*

The notion “in NP” is complicated but hints at an observation in computer science that classes of problems can often be transformed into others. Specifically, problems that are “NP complete” are problems that are NP hard and can be reduced to another problem in NP in polynomial time. That is, solving one of these problems is equivalent to solve another one multiple times, where “multiple” can be expressed as a polynomial. The list of problems that are NP complete is long, including [Boolean satisfiability problem (SAT)](https://en.wikipedia.org/wiki/Boolean_satisfiability_problem), the [Knapsack problem](https://en.wikipedia.org/wiki/Knapsack_problem), the [Hamiltonian path problem, the](https://en.wikipedia.org/wiki/Hamiltonian_path_problem) [Travelling salesman problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem), the [Subgraph isomorphism problem](https://en.wikipedia.org/wiki/Subgraph_isomorphism_problem), [Subset sum problem](https://en.wikipedia.org/wiki/Subset_sum_problem), the [Clique problem](https://en.wikipedia.org/wiki/Clique_problem), the [Vertex cover problem](https://en.wikipedia.org/wiki/Vertex_cover_problem), the [Independent set problem](https://en.wikipedia.org/wiki/Independent_set_problem), the [Dominating set problem](https://en.wikipedia.org/wiki/Dominating_set_problem), and the [Graph coloring problem.](https://en.wikipedia.org/wiki/Graph_coloring_problem) As the reductions from one problem to the other are well known, thoroughly understanding how to solve one problem - namely the TSP - using ant-inspired heuristics will help us to solve many other, related problems.

Closer inspection of the TSP problem reveals that it is actually an assignment problem. There is a tour, which has *M* stations and we need to assign each station with one city so that a global metric is minimized. This is very similar to the *Quadratic Assignment* (QA) problem, which has very general applications. Generally a QA problem provides a set of locations as well as a matrix of flows between them. These can be treatment stations in a hospital that are often frequented in some sequence by patients, goods that need to be moved between machines in a factory, or keys on a keyboard that need to be differently organized as a function of the language that is predominantly typed. In order to be able to apply ant-inspired solution approaches to both TSP and QA as well as other NP complete problems, we need to specify what the general ingredients that make ant colony optimization successful are.

**The Ant Colony Optimization Methaheuristic**

A “heuristic” in an algorithmic context is considered a simple rule that is based on experience, rather than being derived from first principles. It is something that is known to lead to good results quickly, not necessarily the best. For example, A\* search uses a heuristic - usually some kind of distance function - to decide which direction to explore next. While going into the approximate direction of the goal is often successful and should be tried first, it is not always the best action that could have been taken. The idea of depositing artificial pheromones that correspond to the quality of earlier trials to find a good solution and using a non-linear probability function to decide which way to follow are both heuristics. A *meta-heuristic* is a heuristic to build heuristics.

In the quadratic assignment problem, assignments consists of facilities to places. Much like in the TSP problem, we will need a heuristic that helps an ant to initially decide between assignments that are better than others. In the TSP, this heuristic is simply the distance to a city. That is, picking a closer city is preferable to a far-away city when initially constructing a solution. In QA, these heuristics are more complex and known as “distance potential” and “flow potential”. The distance potential of a node is the sum of distances from this node to all other nodes. Nodes with a small distance potential are more central than those with a high distance potential. The flow potential is a facility is the sum of all flows from all other nodes to this node. Intuitively, it makes sense to associate facilities with a flow potentials with nodes that have low distance potentials and the other way round.

We have already shown that the TSP can be reduced to an assignment problem. Each assignment consist of a city and an index when this city needs to be visited during the tour.

In the ant colony metaheuristic, each possible assignment, that is each component of a solution, is associated with a pheromone value. But how can we assign a pheromone value to the information “the n-th city is at the m-th position of the list”? One possible way of doing this is to reduce the solution from a list of cities into a list of edges. As each edge encodes information about where the ant came from, and where it will go, we can now associate a pheromone value with each solution component. You may argue that the search problem has gotten much larger now. After all, there are many more edges in a complete graph then there are nodes, making the alphabet with which we can construct our string much larger. On the other hand, not all combinations are possible anymore as each edge can only start where the last one ended.

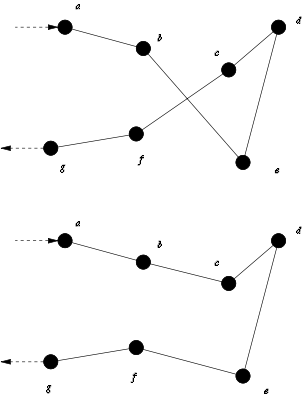
Replacing the solution with a list of edges that ants visit in order rather than maintaining a list of nodes (cities), makes the pheromone deposition analogy straightforward to apply. Calculating the amount of pheromones that need to be deposited is then straightforward: once a complete solution has been constructed, the overall cost can be calculated, and a measure inverse to it - lower cost leading to more pheromone - associated with each solution component associated with this tour.

More generally, the application of ant-like algorithms requires the following:

1. The solution can be broken down into solution components.
2. The overall solution can be evaluated as a whole.
3. “Ants” construct a solution incrementally, randomly picking components using a combination of task-specific heuristic and “pheromones” reflecting previous performance.
4. Pheromones are calculated based on the performance of the overall solution and distributed among solution components.
5. Each iteration can be improved by local search and pruned by pheromone “evaporation” (see below).
6. Multiple iterations of 1-5 until convergence.

*Improvements: evaporation and local search*

There are two more components of the Ant Colony Optimization metaheuristic: pheromone evaporation and local search. Although ants will quickly find a good solution, it is very likely that they get stuck in a local optimum. This happens when not-so-good solutions get reinforced to the point that any other solution becomes very unlikely. This can be countered in two ways. First, pheromones are allowed to evaporate, that is are reduced by a certain percentage at every time step. Initially, this will impede growth of pheromone trails and thereby stimulates exploration. Once a strong path has emerged, however, also evaporation will not help the algorithm to get “unstuck”. Here, performing a “local search” turned out to be tremendously helpful. Local search is the idea of finding improvements by applying small “local” changes to each solution. For example, “2-opt” local search manipulates a solution by swapping two vertices in a solution and inverting the order of vertices in between the two swapped once. An example is shown below. “3-opt” local search instead deletes three edges and reconnect the network in all possible ways. A generalization of  2-opt and 3-opt local search is the Lin-Kernigham heuristic (Lin, Kernigham, 1973), which is adaptive in how many paths between cities need to change at each iteration.



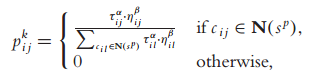
*2-opt local search. The solution* abedcfg *is transformed into* abcdefg *by swapping two entries and inverting the order of the sequence in between. In this case, the inverse of the sequence “d” remains “d”. © PierreSelim CC-SA 3.0*

How local search is implemented is highly dependent on the problem at hand. For example, the TSP requires to inverse paths in between the two assignments that are swapped, but QA does not.

**Ant System (AS), Ant Colony System (ACS) and Max-Min Ant System**

There are multiple variants of Ant Colony Optimization algorithms of which Ant Colony System (ACS) and Max-Min Ant System (MMAS) are the most successful ones (Dorigo, 2006). Being a metaheuristic, you will find a number of similarities between all of them as well as more severe modifications and additions that are driven by specific solution domains.

In all algorithms, a number of *m* ants incrementally creates possible solutions and evaluates their cost, for example the total path length. Let ijbe the the amount of pheromone on the edge between node i and j. Initially, this value can be some constant. Starting from a random node *i*, an ant *k* chooses another node *j* with probability



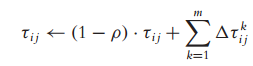
Here N(sp) is the list of feasible solutions that is all edges starting at *i* that have previously not been visited. The probability to chose edge i-j is therefore proportional to its pheromone content ij(amplified by a constant exponent alpha) and a heuristic associated with the assignment i to j, which is amplified by a constant exponent beta. In the TSP, this heuristic is given by the inverse of the distance from city i to city j (dij), given by



In the QAP, the heuristic is the product of node i*’s* distance potential and facility j’s flow potential. The higher this product is, the more desirable such an assignment is. The denominator serves as normalization term by dividing by the sum of pheromone and inverse distance heuristic of all edges that are feasible from i. In order for this to work, ants should start at the node with the lowest distance potential, and then work their way up, however.

The terms alpha and beta now allow to trade between heuristic information and what the algorithm has learned about the problem. If alpha is equal to zero, the probabilities are purely based on distance (with low distances receiving high likelihoods to be picked). If beta is zero, the algorithm ignores distances and fully relies on the pheromone trail. Clearly, there is a trade-off between the two with a positive value of beta leading to quickly find some good solution and a positive value of alpha ensuring that a global optima will not get excluded.

Once a complete solution is created, the amount of  pheromones that ants will deposit is inversely proportional to cost multiplied by a constant. The pheromone on each edge of the graph will then be updated using the following rule



The term (1-)is an evaporation constant, diminishing the amount of pheromone every iteration. The sum now adds the contribution of each *k-th* ant to assignment of  *i* to *j*. In practice, the implementation might be slightly different and require to globally evaporate pheromones in a first step, and retrace the path of every ant and deposit pheromone according to its total cost, in a second step.

Now that the results of first solution have been evaluated and applied to the graph, the ants can be re-started until the best solution does not improve any more.

A widely accepted improvement to the original Ant System algorithm is to limit deposition of pheromones to the best ant and bound both the minimum and maximum value of pheromone. Note that this algorithm gives up on bio-inspiration (by ignoring ants that contribute poor solution components), but is computationally simpler. How to chose appropriate minimum and maximum pheromone values is application specific and (Socha, 2002) provides some guidelines and analysis (Stützle, 2000).

**Summary**

An ant-based solution to a combinatorial optimization problem has the following ingredients:

1. A heuristic which allows to rank order assignments. In the TSP, these are distances to cities, in the QAP these are distance and flow potentials. **(Positive feedback)**
2. A global cost function that can evaluate complete solutions and can be used to deposit pheromones on solution components as a function of their contribution.
3. A probabilistic rule that combines both the heuristic from (1) and past performance information from (2). This rule is usually parameterized, which allows to balance between exploration (from 2) and exploitation (from 1). **(Randomness)**
4. The ability to prune solutions over time using pheromone evaporation. (**Negative Feedback)**
5. Repeating 1-4 as often as necessary until a good-enough solution is found. (**Multiple interactions**)

We see that the Ant-Colony Optimization is much less about shortest path problems than about using the mechanisms from self-organization in order to have a close-to-optimal solution emerge.

**References**

Deneubourg, J. L., Goss, S., Franks, N., & Pasteels, J. M. (1989). The blind leading the blind: modeling chemically mediated army ant raid patterns.Journal of insect behavior, 2(5), 719-725.

Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 26, no. 1 (1996): 29-41.

Dorigo, Marco, Mauro Birattari, and Thomas Stützle. "Ant colony optimization." Computational Intelligence Magazine, IEEE 1, no. 4 (2006): 28-39.

Grassé, Pierre-P. "La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs." Insectes sociaux 6, no. 1 (1959): 41-80.

Goss, Simon, et al. "Self-organized shortcuts in the Argentine ant."Naturwissenschaften 76.12 (1989): 579-581.

[Lin, Shen](https://en.wikipedia.org/w/index.php?title=Shen_Lin&action=edit&redlink=1); [Kernighan, B. W.](https://en.wikipedia.org/wiki/Brian_Kernighan) (1973). "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". Operations Research 21 (2): 498–516. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1287/opre.21.2.498](https://dx.doi.org/10.1287%2Fopre.21.2.498).

Socha, Krzysztof, Joshua Knowles, and Michael Sampels. "A max-min ant system for the university course timetabling problem." In Ant algorithms, pp. 1-13. Springer Berlin Heidelberg, 2002.

Stützle, Thomas, and Holger H. Hoos. "MAX–MIN ant system." Future generation computer systems 16, no. 8 (2000): 889-914.