Ant Colony Optimization

CompSci 760
Patricia J Riddle

Natural Inspiration

The name *Ant Colony Optimization* was chosen to reflect its original inspiration: the foraging behavior of some ant species.

It was inspired by the double-bridge experiment performed by Jean-Louis Deneubourg et al.

Ants are able to find the shortest path to a food source by collectively exploiting pheromones they deposit on the ground while moving.

ACO now includes many components that are no longer related to real ants

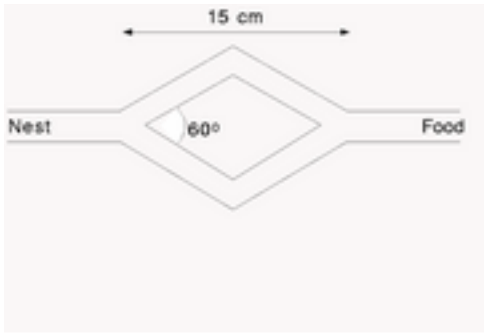
The Double-Bridge Experiment

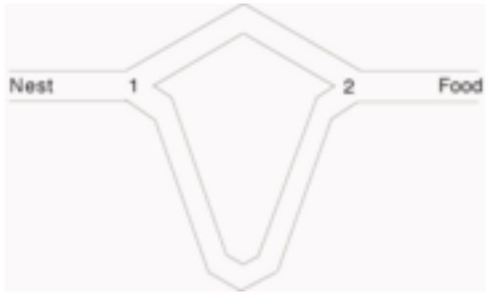
a nest of a colony of *Argentine ants* is connected to a food source by two bridges.

The ants can reach the food source and get back to the nest using any of the two bridges.

The goal of the experiment is to observe the resulting behavior of the colony.

Double Bridges





Bridges with the Same Length

if the two bridges have the same length, the ants tend to converge towards the use of one of the two bridges.

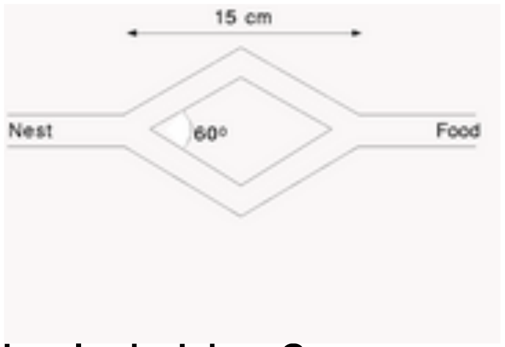
If the experiment is repeated, each of the two bridges is used in about 50% of the cases.

while moving, ants deposit pheromone on the ground;

whenever they must choose which path to follow, their choice is biased by pheromone:

the higher the pheromone concentration found on a particular path, the higher is the probability to follow that path.

Ant Behaviour



How the ants converge towards the use of a single bridge?

At the start the ants explore the surroundings of the nest.

When they arrive at the decision point, they choose probabilistically biased on the pheromone they sense on the two bridges.

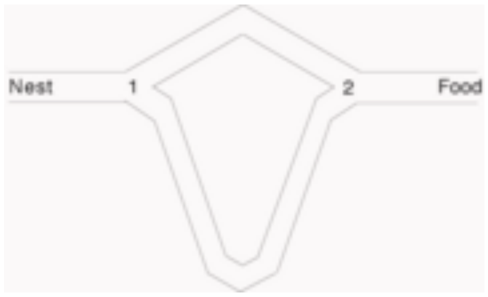
Initially, ants choose one of the two bridges with 50% probability - no pheromone yet.

after some time, because of random fluctuations, one bridge has a higher concentration of pheromone and attracts more ants.

This in turn increases the pheromone level on that bridge, making it more attractive.

It is this **autocatalytic** mechanism that makes the whole colony converge towards the use of the same bridge.

The Short Bridge Experiment



If one of the bridges is significantly shorter, a **second mechanism** plays an important role:

- ants that randomly choose the shorter bridge are the first to reach the food source.
- When these ants, while moving back to the nest, encounter the decision point 2,
  - they sense a higher pheromone on the shorter bridge,
  - they chose that bridge with higher probability and once again it receives additional pheromone.

This fact increases the probability that further ants select it rather than the long one.

Model for Short Bridge

Goss et al. (1989) developed a model of the observed behavior:

assuming that at a given moment in time m_1 ants have used the first bridge and m_2 the second one,

the probability for an ant to choose the first bridge is:

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h}$$

where parameters k and h are to be fitted to the experimental data---obviously, $p_2 = 1 - p_1$.

Monte Carlo simulations showed a very good fit for $k \approx 20$ and $h \approx 2$ (Goss et al. 1989).

It is this equation that inspired the equation used in ant system, the first ACO algorithm.

Monte Carlo

Monte Carlo methods are a class of computational algorithms that rely on **repeated random sampling** to compute results.

often used when simulating physical and mathematical systems.

rely on **repeated computation** and **random or pseudo-random numbers**, so are suited to computer calculation.

used when unfeasible or **impossible to compute an exact result** with deterministic algorithm.

Monte Carlo Pattern

No single Monte Carlo method;

large and widely-used class of approaches.

approaches tend to follow a particular pattern:

- Define a domain of possible inputs.
- Generate inputs randomly from the domain.
- Perform a deterministic computation using the inputs.
- Aggregate the results of the individual computations into the final result.

Monte Carlo Example

Approximate the value of π using a Monte Carlo method:

- Draw a square, inscribe a circle within it.
  - The ratio of the area of an inscribed circle to that of the surrounding square is π/4.
- Uniformly scatter some objects of uniform size throughout the square.
- The objects should fall in the areas in approximately the same ratio.
- Counting the number of objects in the circle and dividing by the total number of objects in the square will yield an approximation for π/4.
- Multiplying the result by 4 will then yield an approximation for π itself.

Ant Colony Overview

Ant colony optimization is a population-based **metaheuristic** that can be used to find approximate solutions to difficult combinatorical optimization problems.

In ant colony optimization (ACO), a set of software agents called "artificial ants" search for good solutions to a given optimization problem transformed into the problem of finding the **minimum cost path on a weighted graph**.

The artificial ants incrementally build solutions by moving on the graph.

The **solution construction process is stochastic** and is biased by a **pheromone model** - a set of parameters associated with graph components (either nodes or edges) the values of which are modified at runtime by the ants.

Metaheuristic

In computer science, metaheuristic designates a computational method that optimizes a problem by **iteratively** trying to **improve** a **candidate solution** with regard to a given measure of quality.

Metaheuristics make **few or no assumptions** about the problem being optimized and can search very large spaces of candidate solutions.

However, metaheuristics **do not guarantee an optimal solution** is ever found.

Many metaheuristics implement some form of **stochastic optimization**.

Combinatorical Optimization

The goal of combinatorial optimization is to find a **discrete mathematical object** (such as a bit string or permutation) that maximizes (or minimizes) an **arbitrary function** specified by the user of the metaheuristic.

These objects are generically called *states, and the set of all candidate states is the search space.*

The nature of the states and the search space are usually problem-specific.

Metaheurstic vs Combinatorical Optimization

A metaheuristic uses combinatorical optimization (or straight optimization if it is not discrete – Particle Swarm Optimization) to **alter/learn** the heuristic while the search is conducted (hence the “meta”).

Metaheuristic vs Heuristic

Metaheuristics change over time as the algorithm runs its iterations. (phermone)

Heuristics are static through out a search (inverse distance between towns in TSP)

What is Optimisation

optimization, or **mathematical programming**, refers to the study of problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set.

An optimization problem can be represented in the following way

Given: a function $f: A \to \Re$ from some set A to the real numbers

Sought: an element x_0 in A

such that $f(x_0) \leq f(x)$ for all x in A ("minimization") or

such that $f(x_0) \geq f(x)$ for all x in A ("maximization").

ACO Successes

applied successfully to many classical combinatorial optimization problems.

- routing in communication networks
- stochastic version of well-known combinatorial optimization problem, such as the probabilistic traveling salesman problem.

ACO has been extended so that it can be used to solve continuous and mixed-variable optimization problems (Socha and Dorigo 2004).

Ant colony optimization is probably the most successful example of artificial/engineering swarm intelligence system

Evaporation

In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails.

If other ants find such a path, they are likely not to keep traveling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food.

Over time, however, the pheromone trail **starts to evaporate**, thus reducing its attractive strength.

The **more time it takes for an ant to travel down the path** and back again, the **more time the pheromones have to evaporate**.

Evaporation Avoids Local Optima

A short path, gets marched over faster, thus the pheromone density remains high - it is laid on the path as fast as it can evaporate.

Pheromone evaporation has also the advantage of **avoiding the convergence to a locally optimal solution**.

If there were **no evaporation**, the paths chosen by the first ants would tend to be excessively attractive to the following ones.

The exploration of the solution space would be constrained.

Positive Feedback

When one ant finds a good (i.e. short) path from the colony to a food source, other ants are more likely to follow that path

positive feedback eventually leads all the ants following a single path.

ant colony algorithm mimics this behavior with "simulated ants" walking around the graph representing the problem to solve.

ACO Dynamic Advantage

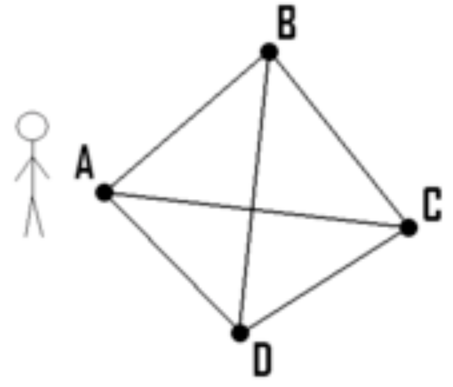
Ant colony optimization algorithms can produce near-optimal solutions to the traveling salesman problem.

They have an **advantage over simulated annealing** and genetic algorithm approaches when the graph may change dynamically;

- the ant colony algorithm can be **run continuously** and **adapt to changes in real time**.

This is of interest in network routing and urban transportation systems.

TSP example



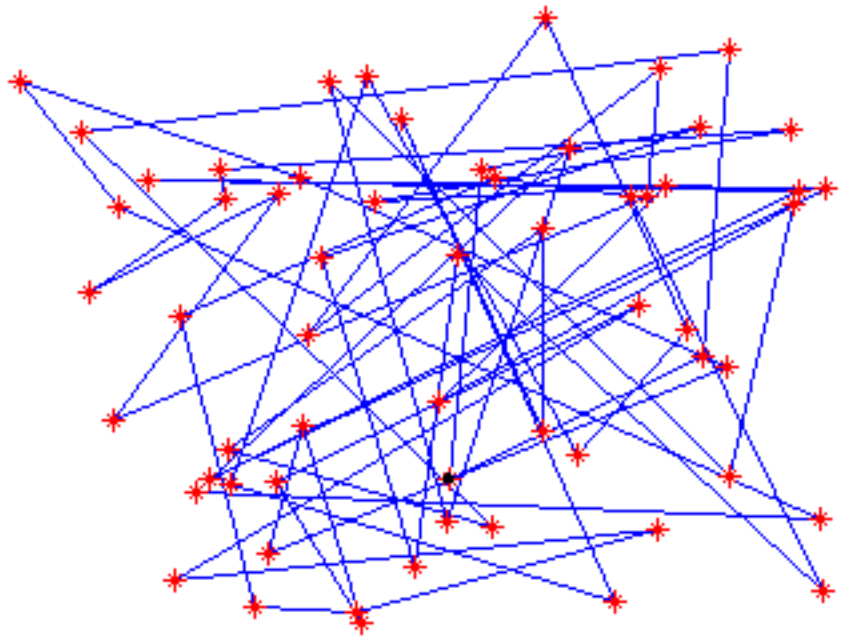
In the traveling salesman problem (TSP) a set of locations (cities) and the distances between them are given.

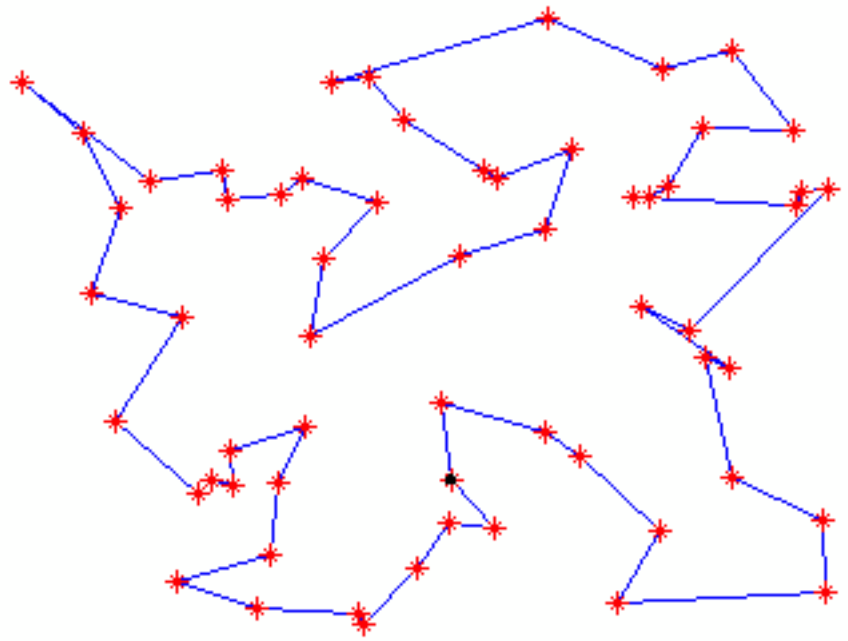
The problem consists of finding a **closed tour of minimal length** that visits each city once and only once.

To apply ACO to the TSP, we consider the graph defined by associating the set of cities with the set of vertices of the graph.

This graph is called ***construction graph***.

Larger TSP example





Construction Graph

the construction graph is fully connected and the number of vertices is equal to the number of cities.

the lengths of the edges between the vertices are proportional to the distances between the cities represented by these vertices

we associate **pheromone values** and **heuristic values** with the **edges of the graph**.

Pheromone values are **modified at runtime** and represent the cumulated experience of the ant colony,

heuristic values are problem dependent values that, in the case of the TSP, are the inverse of lengths of the edges.

The Solution

The ants construct the solutions as follows.

- Each ant starts from a randomly selected city (vertex of the construction graph).
- Then, at each construction step it moves along the edges of the graph.
- Each ant keeps a memory of its path, and in subsequent steps it chooses among the edges that do not lead to vertices that it has already visited.
- An ant has constructed a solution once it has visited all the vertices of the graph.

Probabilistic Rule

At each construction step,

- an ant probabilistically chooses the edge to follow among those that lead to yet unvisited vertices.

The **probabilistic rule** is biased by pheromone values and heuristic information:

- the **higher the pheromone** and the **heuristic value** associated to an edge, the **higher the probability** an **ant will choose** that particular **edge**.

Pheromone Update

Once all the ants have completed their tour, the pheromone on the edges is updated.

Each of the pheromone values is initially **decreased by a certain percentage**. - **Evaporation**

Each edge then receives an amount of **additional pheromone proportional** to the **quality of the solutions** to which it belongs (there is one solution per ant).

This procedure is repeatedly applied until a termination criterion is satisfied.

Formal Definition of a Combinatorial Optimization Problem

The first step for the application of ACO to a combinatorial optimization problem (COP) consists in defining a model of the COP as a triplet *(S,Ω,f)*, where:

- *S* is a search space defined over a finite set of discrete decision variables;

- *Ω* is a set of constraints among the variables; and

- $f : S \rightarrow \Re_0^+$ is an objective function to be minimized
  - as maximizing over f is the same as minimizing over -f, every COP can be described as a minimization problem.

Search Space Definition

The search space *S* is defined as follows.

- A set of discrete variables X_i, $i = 1 \ldots n$, with values, $v_i^j \in D_i = \{v_i^1, \ldots, v_i^{|D_i|}\}$ is given.

- Elements of *S* are full assignments, that is, assignments in which each variable X_i has a value v_i^j assigned from its domain .

- The set of feasible solutions S_Ω is given by the elements of *S* that satisfy all the constraints in the set Ω.

Global optima

A solution $s^* \in S_\Omega$ is called a global optimum if and only if: $f(s^*) \leq f(s) \forall s \in S_\Omega$

The set of all globally optimal solutions is denoted by $S_\Omega^* \subseteq S_\Omega$.

Solving a COP requires finding at least one $s^* \in S_\Omega^*$

Ant colony metaheuristic

ants build a solution to a combinatorial optimization problem by traversing a fully connected construction graph

each instantiated decision variable $X_i=v_i^j$ is called a *solution component* and denoted by c_{ij}.

The set of all possible solution components is *C*.

Then the construction graph $G_C(V,E)$ is defined by associating the components *C* either with the set of vertices *V* or with the set of edges *E*.

Pheromone Trails

A pheromone value τ_{ij} is associated with each component c_{ij}.

(Note pheromone values are generally a function of the algorithm's iteration *t:* $\tau_{ij}=\tau_{ij}(t)$.)

Pheromone values allow the **probability distribution** of different components of the solution to be **modeled**.

Pheromone values are **used and updated** by the ACO algorithm during search.

The Ant Moves

The **ants move** from vertex to vertex along the edges of the construction graph **exploiting** information provided by the **pheromone values** and incrementally building a solution.

Additionally, the **ants deposit** a certain amount of **pheromone** on the components, that is, either **on the vertices or on the edges** that they traverse.

The **amount** $\Delta\tau$ of pheromone deposited may depend on the **quality of the solution** found.

Subsequent ants utilize the pheromone information as a guide towards more promising regions of the search space.

ACO Metaheuristic

```
Set parameters, initialize phermone trails
SCHEDULE_ACTIVITES
  ConstructAntSolutions
  DaemonActions       (optional)
  UpdatePheromones
END_SCHEDULE_ACTIVITIES
```

The Metaheuristic

The metaheuristic consists of an initialization step and of three algorithmic components whose activation is regulated by the Schedule_Activities construct.

This construct is repeated until a **termination criterion** is met.

Typical criteria are a **maximum number of iterations** or a **maximum CPU time**.

Schedule Activities

The Schedule_Activities construct does not specify how the three algorithmic components are scheduled and synchronized.

In most applications of ACO to NP-hard problems however, the three algorithmic components undergo a loop that consists in

1. the construction of solutions by all ants,
2. the (optional) improvement of these solution via the use of a local search algorithm, and
3. the update of the pheromones.

Construct Ant Solutions

A set of m artificial ants construct solutions from elements of a finite set of available solution components $C=\{c_{ij}\}$, $i=1,\ldots,n$, $j=1,\ldots,|D_i|$.

A solution construction starts with an empty partial solution $s^p=\emptyset$.

At each construction step, the current partial solution s^p is extended by adding a feasible solution component from the set of feasible neighbors $N(s^p)\subseteq C$.

The process of constructing solutions can be regarded as a path on the construction graph $G_C(V,E)$.

The allowed paths in G_C are implicitly defined by the solution construction mechanism that defines the set $N(s^p)$ with respect to a partial solution s^p.

Choosing Solution Components

The choice of a solution component from $N(s^p)$ is done probabilistically at each construction step.

The exact rules for the probabilistic choice of solution components vary across different ACO variants.

The best known rule is the one of ant system (AS) (Dorigo et al. 1991, 1996):

$$p(c_{ij} \mid s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \forall c_{ij} \in N(s^p)$$

where τ_{ij} and η_{ij} are the pheromone value and the heuristic value associated with the component .

α and β are positive real parameters whose values determine the relative importance of pheromone versus heuristic information.

Daemon Actions

Once solutions have been constructed, and before updating the pheromone values, problem specific actions may be required.

Often called *daemon actions*, they can be used to implement problem specific and/or centralized actions, which cannot be performed by single ants.

The most used daemon action is the application of **local search** to the constructed solutions: the locally optimized solutions are then used to decide which pheromone values to update.

Pheromone Update

The aim of the pheromone update is to

increase the pheromone values associated with **good** solutions, and

decrease those associated with **bad** ones.

Usually, this is achieved

1. by **decreasing all** the pheromone values through ***pheromone evaporation***, and
2. by **increasing** the pheromone levels associated with a chosen set of **good solutions** S_{upd}:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot \sum_{s\in S_{upd} \mid c_{ij}\in s} F(s)$$

where S_{upd} is the set of solutions that are used for the update, $\rho\in(0,1]$ is a parameter called evaporation rate, and $F : S \rightarrow R_0^+$ is a function such that

$$f(s) < f(s') \Rightarrow F(s) \geq F(s'), \forall s \neq s' \in S$$

F(·) is commonly called the *fitness function*.

Pheromone Evaporation

Pheromone evaporation implements a useful form of ***forgetting***, favoring the **exploration of new areas** in the search space.

Different ACO algorithms, ant colony system (ACS) or MAX-MIN ant system (MMAS), differ in the way they update the pheromone.

Instantiations of the update rule given above are obtained by different specifications of S_{upd}, which in many cases is a subset of $S_{iter} \cup \{S_{bs}\}$, where

S_{iter} is the set of solutions that were constructed in the **current iteration**, and

S_{bs} is the ***best-so-far*** **solution**, that is, the best solution found since the first algorithm iteration.

A well-known example is the **AS-update rule**, that is, the update rule of ant system (Dorigo et al. 1991, 1996): $S_{upd} \leftarrow S_{iter}$

Iteration Best Updates

An example of a pheromone update rule that is more often used in practice is the **IB-update rule** (where IB stands for *iteration-best*):

$$S_{upd} \leftarrow \arg\max_{s \in S_{iter}} F(s)$$

The IB-update rule introduces a **much stronger bias** towards the good solutions found than the AS-update rule.

Although this **increases the speed with which good solutions are found**, it also **increases the probability of premature convergence**.

Best So Far Updates

An **even stronger bias** is introduced by the **BS-update rule**, where BS refers to the use of the best-so-far solution s_{bs}.

In this case, s_{upd} is set to $\{s_{bs}\}$.

In practice, ACO algorithms that **use variations of the IB-update or the BS-update rules** and that additionally **include mechanisms to avoid premature convergence**, achieve better results than those that use the AS-update rule.

Main Types of ACO

Several special cases of the ACO metaheuristic have been proposed in the literature.

overview, in the historical order in which they were introduced, the three most successful ones:

- **ant system (AS)** (Dorigo 1992, Dorigo et al. 1991, 1996),
- **ant colony system (ACS)** (Dorigo & Gambardella 1997), and
- **MAX-MIN ant system (MMAS)** (Stützle & Hoos 2000).

In order to illustrate the differences between them clearly, we use the the traveling salesman problem.

Ant System

Ant system (AS) was the first ACO algorithm to be proposed in the literature (Dorigo et al. 1991, Dorigo 1992, Dorigo et al. 1996).

Its main characteristic is that the **pheromone values** are **updated** by ***all* the ants** that have completed the tour.

Solution components c_{ij} are the edges of the graph, and the pheromone update for τ_{ij}, that is, for the pheromone associated to the edge joining cities *i* and *j*, is performed as follows:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$$

where $\rho \in (0,1]$ is the evaporation rate,

m is the number of ants, and

$\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on edge *(i,j)* by the *k*-th ant

$$\Delta\tau_{ij}^{k} = \left\langle \begin{array}{l} \dfrac{1}{d_{ij}}\ if\ ant\ k\ used\ edge\ (i,j)\ in\ its\ tour, \\ 0 \quad otherwise, \end{array} \right.$$

where d_{ij} is the length of edge from i to j.

Ant System Continued

When constructing the solutions, the ants in AS traverse the construction graph and make a probabilistic decision at each vertex.

The transition probability $p(c_{ij}|s_k^p)$ of the k-th ant moving from city i to city j is given by:

$$P(c_{ij} \mid s_k^p) = \left\langle \begin{array}{ll} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s_k^p)} \tau_{il}^{\alpha} \eta_{il}^{\beta}} & if \;\; j \in N(s_k^p) \\ 0 & otherwise \end{array} \right.$$

where

$N(s_k^p)$ is the set of components that do not belong yet to the partial solution s_k^p of ant k, and

α and β are parameters that control the relative importance of the pheromone versus the heuristic information

Ant Colony System

The first major improvement over the original ant system to be proposed was ant colony system (ACS), introduced by Dorigo and Gambardella (1997).

The first important difference between ACS and AS is the form of the decision rule used by the ants during the construction process.

Ants in ACS use the so-called ***pseudorandom proportional* rule**:

- the probability for an ant to move from city *i* to city *j* depends on a random variable *q* uniformly distributed over *[0,1]*, and a parameter q_0;
- if $q \leq q_0$, then, among the feasible components, the component that maximizes the product $\tau_{il}\eta_{il}^{\beta}$ is chosen, otherwise the same equation as AS is used.

Diversity Component

This rather **greedy rule**, which **favors exploitation** of the pheromone information, is **counterbalanced** by the introduction of a diversifying component: the ***local pheromone update***.

The local pheromone update is performed by all ants after each construction step.

Each ant applies it only to the last edge traversed:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$$

where $\rho \in (0,1]$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone.

Local Update

The main goal of the local update is to **diversify the search performed by subsequent ants during one iteration**.

In fact, decreasing the pheromone concentration on the edges as they are traversed during one iteration **encourages subsequent ants to choose other edges** and hence to **produce different solutions**.

This makes it less likely that several ants produce identical solutions during one iteration.

Additionally, because of the local pheromone update in ACS, the **minimum values of the pheromone are limited**.

Offline Pheromone Update

As in AS, also in ACS at the end of the construction process a pheromone update, called ***offline* pheromone update**, is performed.

ACS offline pheromone update is **performed only by the best ant**, that is, only edges that were visited by the best ant are updated, according to the equation:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot\Delta\tau_{ij}^{best}$$

where $\Delta\tau_{ij}^{best}=1/L_{best}$ if the best ant used *edge(i,j)* in its tour,
$\Delta\tau_{ij}^{best}=0$ otherwise

L_{best} can be set to

iteration-best, L_{ib} - the length of the best tour found in the current iteration or

best-so-far, L_{bs} – the best solution found since the start of the algorithm.

Most of the innovations introduced by ACS were introduced first in Ant-Q, a preliminary version of ACS by the same authors.

Max-Min Ant System

MAX-MIN ant system (MMAS) is another improvement, proposed by Stützle and Hoos (2000), over the original ant system idea.

MMAS differs from AS in that

1. **only the best ant adds pheromone trails**, and
2. the **minimum and maximum values of the pheromone are explicitly limited**

   in AS and ACS these values are limited implicitly, that is, the value of the limits is a result of the algorithm working rather than a value set explicitly by the algorithm designer.

Phermone Update

The pheromone update equation takes the following form:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot\Delta\tau_{ij}^{best}$$

where $\Delta\tau_{ij}^{best}=1/L_{best}$ if the best ant used *edge(i,j)* in its tour,
$\Delta\tau_{ij}^{best}=0$ otherwise
L_{best} is the length of the tour of the best ant.

As in ACS, L_{best} may be set (subject to the algorithm designer decision) either to L_{ib} or to L_{bs}, or to a combination of both.

The pheromone values are constrained between τ_{min} and τ_{max} by verifying, after they have been updated by the ants, that all pheromone values are within the imposed limits:

τ_{ij} is set to τ_{max} if $\tau_{ij} > \tau_{max}$ and to
τ_{min} if $\tau_{ij} < \tau_{min}$.

The **pheromone update** equation of MMAS is applied, as it is in the case for AS, **to all the edges** while in ACS it is applied only to the edges visited by the best ants.

MIN and MAX values

The minimum value τ_{min} is most often experimentally chosen (however, some theory about how to define its value analytically has been developed in (Stützle & Hoos 2000)).

The maximum value τ_{max} may be calculated analytically provided that the optimum ant tour length is known.

In the case of the TSP, $\tau_{max}=1/(\rho \cdot L^*)$, where L^* is the length of the optimal tour.

If L^* is not known, it can be approximated by L_{bs}.

It is also important to note that **the initial value of the trails is set to τ_{max}**, and that the algorithm is **restarted when no improvement** can be observed for a **given number of iterations**.

Traveling tournament Problem

Double round robin (A at B and B at A): n teams need 2*n-2 slots.

No more than three consecutive home or three consecutive road games for any team

No repeaters (A at B, followed immediately by B at A)

Defined in 2001

Only solved for 8 teams or less

David Uthus' Work

Ant Colony algorithm based on Max-Min does almost as well as simulated annealing – the current best approach

IDA* technique

- found (in significantly less time) previously known optimal solutions
- found optimal solutions where optimal solutions were unknown (for 10 teams!!!)
- found better (lower bound) solutions where only lower bounds are known.

How is ACO like Reinforcement learning?

- Exploration vs exploitation
- Try many “random” solutions
- Update numbers and iterate to a solution

When should you do ACO

- When a solution is easy to find, but
- A good solution is hard to find, and
- There is a gradient on better and better solutions

Can you train an AC with Stochastic Gradient Descent?

- SGD in the space of pheromones
- Acceleration techniques for ACO based on gradient based reinforcement learning

Applications of ACO

The initial applications of ACO were in the domain of NP-hard combinatorial optimization problems.

The largest body of ACO research is still, not surprisingly, to be found in this area.

Complete overview of these applications in (Dorigo & Stützle 2004).

Another application that was considered early in the history of ACO is routing in telecommunication networks.

A particularly successful example of ACO algorithm in this domain is AntNet (Di Caro & Dorigo 1998).

Current ACO Trends

Current research in ACO algorithms is devoted both to the development **of theoretical foundations** and to the application of the metaheuristic to new challenging problems.

The development of a theoretical foundation was started by Gutjahr, who was the first to prove convergence in probability of an ACO algorithm (Gutjahr 2000).

An overview of theoretical results available for ACO can be found in (Dorigo & Blum 2005).

Concerning applications, the use of ACO for the solution of **dynamic, multiobjective, stochastic, continuous and mixed-variable optimization** problems is a current hot topic, as well as the creation of parallel implementations capable of taking advantage of the new available parallel hardware.

Many papers reporting on current research can be found in the proceedings of the ANTS conference or in the Swarm Intelligence journal.