# Introduction to Ant Colony Optimization

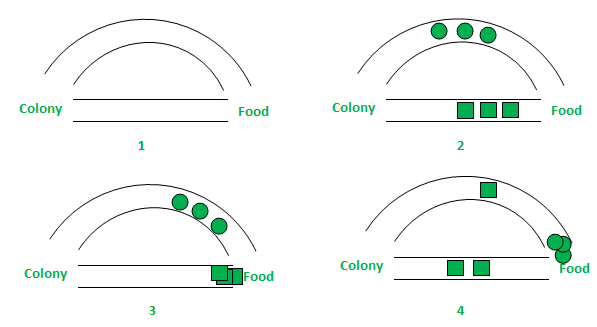
* **Difficulty Level :** [Hard](https://www.geeksforgeeks.org/hard/)
* **Last Updated :** 17 May, 2020

The algorithmic world is beautiful with multifarious strategies and tools being developed round the clock to render to the need for high-performance computing. In fact, when algorithms are inspired by natural laws, interesting results are observed. Evolutionary algorithms belong to such a class of algorithms. These algorithms are designed so as to mimic certain behaviours as well as evolutionary traits of the human genome. Moreover, such algorithmic design is not only constrained to humans but can be inspired by the natural behaviour of certain animals as well. The basic aim of fabricating such methodologies is to provide realistic, relevant and yet some low-cost solutions to problems that are hitherto unsolvable by conventional means.

Different optimization techniques have thus evolved based on such evolutionary algorithms and thereby opened up the domain of metaheuristics. **Metaheuristic** has been derived from two Greek words, namely, **Meta** meaning **one level above** and **heuriskein** meaning **to find**. Algorithms such as the Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are examples of swarm intelligence and metaheuristics. The goal of swarm intelligence is to design intelligent multi-agent systems by taking inspiration from the collective behaviour of social insects such as ants, termites, bees, wasps, and other animal societies such as flocks of birds or schools of fish.

**Background:**

Ant Colony Optimization technique is purely inspired from the **foraging** behaviour of ant colonies, first introduced by Marco Dorigo in the 1990s. Ants are eusocial insects that prefer community survival and sustaining rather than as individual species. They communicate with each other using sound, touch and pheromone. **Pheromones** are organic chemical compounds secreted by the ants that trigger a social response in members of same species. These are chemicals capable of acting like hormones outside the body of the secreting individual, to impact the behaviour of the receiving individuals. Since most ants live on the ground, they use the soil surface to leave pheromone trails that may be followed (smelled) by other ants.  
Ants live in community nests and the underlying principle of ACO is to observe the movement of the ants from their nests in order to search for food in the shortest possible path. Initially, ants start to move randomly in search of food around their nests. This randomized search opens up multiple routes from the nest to the food source. Now, based on the quality and quantity of the food, ants carry a portion of the food back with necessary pheromone concentration on its return path. Depending on these pheromone trials, the probability of selection of a specific path by the following ants would be a guiding factor to the food source. Evidently, this probability is based on the concentration as well as the rate of evaporation of pheromone. It can also be observed that since the evaporation rate of pheromone is also a deciding factor, the length of each path can easily be accounted for.

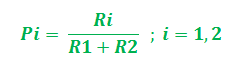


In the above figure, for simplicity, only two possible paths have been considered between the food source and the ant nest. The stages can be analyzed as follows:

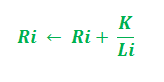
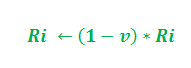
1. **Stage 1:**All ants are in their nest. There is no pheromone content in the environment. (For algorithmic design, residual pheromone amount can be considered without interfering with the probability)
2. **Stage 2:**Ants begin their search with equal (0.5 each) probability along each path. Clearly, the curved path is the longer and hence the time taken by ants to reach food source is greater than the other.
3. **Stage 3:**The ants through the shorter path reaches food source earlier. Now, evidently they face with a similar selection dilemma, but this time due to pheromone trail along the shorter path already available, probability of selection is higher.
4. **Stage 4:**More ants return via the shorter path and subsequently the pheromone concentrations also increase. Moreover, due to evaporation, the pheromone concentration in the longer path reduces, decreasing the probability of selection of this path in further stages. Therefore, the whole colony gradually uses the shorter path in higher probabilities. So, path optimization is attained.

**Algorithmic Design:**

Pertaining to the above behaviour of the ants, an algorithmic design can now be developed. For simplicity, a single food source and single ant colony have been considered with just two paths of possible traversal. The whole scenario can be realized through weighted graphs where the ant colony and the food source act as vertices (or nodes); the paths serve as the edges and the pheromone values are the weights associated with the edges.  
Let the graph be **G = (V, E)** where V, E are the edges and the vertices of the graph. The vertices according to our consideration are **Vs** (Source vertex – ant colony) and **Vd** (Destination vertex – Food source), The two edges are **E1** and **E2** with lengths **L1** and **L2** assigned to each. Now, the associated pheromone values (indicative of their strength) can be assumed to be **R1** and **R2** for vertices E1 and E2 respectively. Thus for each ant, the starting probability of selection of path (between E1 and E2) can be expressed as follows:



Evidently, if R1>R2, the probability of choosing E1 is higher and vice-versa. Now, while returning through this shortest path say Ei, the pheromone value is updated for the corresponding path. The updation is done based on the length of the paths as well as the evaporation rate of pheromone. So, the update can be step-wise realized as follows:

1. **In accordance to path length** –  
     
   In the above updation, i = 1, 2 and ‘K’ serves as a parameter of the model. Moreover, the update is dependent on the length of the path. Shorter the path, higher the pheromone added.
2. **In accordance to evaporation rate of pheromone** –  
     
   The parameter ‘v’ belongs to interval (0, 1] that regulates the pheromone evaporation. Further, i = 1, 2.

At each iteration, all ants are placed at source vertex Vs (ant colony). Subsequently, ants move from Vs to Vd (food source) following step 1. Next, all ants conduct their return trip and reinforce their chosen path based on step 2.

**Pseudocode:**

**Procedure** AntColonyOptimization:

Initialize necessary parameters and pheromone trials;

**while** not termination **do**:

Generate ant population;

Calculate fitness values associated with each ant;

Find best solution through selection methods;

Update pheromone trial;

**end** while

**end** procedure

The pheromone update and the fitness calculations in the above pseudocode can be found through the step-wise implementations mentioned above.  
Thus, the introduction of the ACO optimization technique has been established. The application of the ACO can be extended to various problems such as the famous **TSP (Travelling Salesman Problem)**.

**References:**  
https://www.ics.uci.edu/~welling/teaching/271fall09/antcolonyopt.pdf

The first ant colony optimisation algorithm was introduced by [Marco Dorigo](http://iridia.ulb.ac.be/~mdorigo/HomePageDorigo/) in the report Positive Feedback as a Search Strategy (1991) and his PhD thesis Optimization, Learning and Natural Algorithms (1992). He's still one of the leading figures in the field of [swarm intelligence](https://en.wikipedia.org/wiki/Swarm_intelligence) (having also written or co-written several papers and books). Another important person that contributed to ACO algorithms is [Luca Gambardella](http://people.idsia.ch/~luca/) (co-director of [IDSIA](http://www.idsia.ch/)).

There are several ACO algorithms. They are all based on the way real ants behave, that is, by leaving a substance called "pheromone" on the ground in order to communicate. More specifically, the amount of pheromone is associated with value (e.g. food): more pheromone means more value. (It should now be clear the reason behind the queues real ants form).

A list of ACO algorithms can be found at [http://iridia.ulb.ac.be/~mdorigo/ACO/publications.html](http://iridia.ulb.ac.be/~mdorigo/ACO/publications.html#THIRD). For reproducibility, here's a (non-exhaustive) list:

* Ant System
* Elitist Ant System
* Ant-Q
* Ant Colony System
* Max-Min Ant System
* Rank-based Ant System
* ANTS
* Hyper Cube - ACO

ACO algorithms have been applied to [**combinatorial**](https://en.wikipedia.org/wiki/Combinatorial_optimization) and [**NP-complete**](https://en.wikipedia.org/wiki/NP-completeness) (e.g. the [travelling salesman problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem)) problems. ACO algorithms are thus a collection of meta-heuristic and probabilistic algorithms (in the same family of [simulated annealing](https://en.wikipedia.org/wiki/Simulated_annealing)) to tackle often considered [intractable problems](https://en.wikipedia.org/wiki/Computational_complexity_theory#Intractability). The [related Wikipedia article](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#Applications) contains a more exhaustive section dedicated to the applications of these algorithms. ACO algorithms are often combined with [local search algorithms](https://en.wikipedia.org/wiki/Local_search_(optimization)) (like the 2-opt or 3-opt).

I would suggest you to start with the travelling salesman problem, which was the first application of these algorithms. You can have a look at the reference implementations at <http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html>, where you can also find software to solve specific tasks (not just the TSP, such as maximum clique problems).

Ant colony optimization algorithms

In [computer science](https://en.wikipedia.org/wiki/Computer_science) and [operations research](https://en.wikipedia.org/wiki/Operations_research), the **ant colony optimization** [algorithm](https://en.wikipedia.org/wiki/Algorithm) (**ACO**) is a [probabilistic](https://en.wikipedia.org/wiki/Probability) technique for solving computational problems which can be reduced to finding good paths through [graphs](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)). Artificial ants stand for [multi-agent](https://en.wikipedia.org/wiki/Multi-agent) methods inspired by the behavior of real [ants](https://en.wikipedia.org/wiki/Ant). The pheromone-based communication of biological ants is often the predominant paradigm used.[[2]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-2) Combinations of artificial ants and [local search](https://en.wikipedia.org/wiki/Local_search_(optimization)) algorithms have become a method of choice for numerous optimization tasks involving some sort of [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)), e.g., [vehicle routing](https://en.wikipedia.org/wiki/Vehicle_routing_problem) and internet [routing](https://en.wikipedia.org/wiki/Routing).

As an example, ant colony optimization[[3]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-3) is a class of [optimization](https://en.wikipedia.org/wiki/Optimization_(computer_science)) [algorithms](https://en.wikipedia.org/wiki/Algorithm) modeled on the actions of an [ant colony](https://en.wikipedia.org/wiki/Ant_colony).[[4]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-Birattari_Pellegrini_Dorigo_2007_pp._732%E2%80%93742-4) Artificial 'ants' (e.g. simulation agents) locate optimal solutions by moving through a [parameter space](https://en.wikipedia.org/wiki/Parameter_space) representing all possible solutions. Real ants lay down [pheromones](https://en.wikipedia.org/wiki/Pheromone) directing each other to resources while exploring their environment. The simulated 'ants' similarly record their positions and the quality of their solutions, so that in later simulation iterations more ants locate better solutions.[[5]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-5) One variation on this approach is [the bees algorithm](https://en.wikipedia.org/wiki/Bees_algorithm), which is more analogous to the foraging patterns of the [honey bee](https://en.wikipedia.org/wiki/Honey_bee), another social insect.

This algorithm is a member of the ant colony algorithms family, in [swarm intelligence](https://en.wikipedia.org/wiki/Swarm_intelligence) methods, and it constitutes some [metaheuristic](https://en.wikipedia.org/wiki/Metaheuristic) optimizations. Initially proposed by [Marco Dorigo](https://en.wikipedia.org/wiki/Marco_Dorigo) in 1992 in his PhD thesis,[[6]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-6)[[7]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-M._Dorigo,_Optimization,_Learning_and_Natural_Algorithms-7) the first algorithm was aiming to search for an optimal path in a graph, based on the behavior of [ants](https://en.wikipedia.org/wiki/Ants) seeking a path between their [colony](https://en.wikipedia.org/wiki/Ant_colony) and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants. From a broader perspective, ACO performs a model-based search[[8]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms" \l "cite_note-8) and shares some similarities with [estimation of distribution algorithms](https://en.wikipedia.org/wiki/Estimation_of_distribution_algorithm).

## **Overview[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=1" \o "Edit section: Overview)]**

In the natural world, ants of some species (initially) wander [randomly](https://en.wikipedia.org/wiki/Random), and upon finding food return to their colony while laying down [pheromone](https://en.wikipedia.org/wiki/Pheromone) trails. If other ants find such a path, they are likely not to keep travelling at random, but instead to follow the trail, returning and reinforcing it if they eventually find food (see [Ant communication](https://en.wikipedia.org/wiki/Ant#Communication)).[[9]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-9)

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. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained. The influence of pheromone evaporation in real ant systems is unclear, but it is very important in artificial systems.[[10]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-10)

The overall result is that 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, and [positive feedback](https://en.wikipedia.org/wiki/Positive_feedback) eventually leads to many ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.

### Ambient networks of intelligent objects**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=2" \o "Edit section: Ambient networks of intelligent objects)]**

New concepts are required since “intelligence” is no longer centralized but can be found throughout all minuscule objects. Anthropocentric concepts have been known to lead to the production of IT systems in which data processing, control units and calculating forces are centralized. These centralized units have continually increased their performance and can be compared to the human brain. The model of the brain has become the ultimate vision of computers. [Ambient networks](https://en.wikipedia.org/wiki/Ambient_networks) of intelligent objects and, sooner or later, a new generation of information systems which are even more diffused and based on nanotechnology, will profoundly change this concept. Small devices that can be compared to insects do not dispose of a high intelligence on their own. Indeed, their intelligence can be classed as fairly limited. It is, for example, impossible to integrate a high performance calculator with the power to solve any kind of mathematical problem into a biochip that is implanted into the human body or integrated in an intelligent tag which is designed to trace commercial articles. However, once those objects are interconnected they dispose of a form of intelligence that can be compared to a colony of ants or bees. In the case of certain problems, this type of intelligence can be superior to the reasoning of a centralized system similar to the brain.[[11]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-Waldner_2008_214-11)

Nature offers several examples of how minuscule organisms, if they all follow the same basic rule, can create a form of [collective intelligence](https://en.wikipedia.org/wiki/Collective_intelligence) on the macroscopic level. Colonies of social insects perfectly illustrate this model which greatly differs from human societies. This model is based on the co-operation of independent units with simple and unpredictable behavior.[[12]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-12) They move through their surrounding area to carry out certain tasks and only possess a very limited amount of information to do so. A colony of ants, for example, represents numerous qualities that can also be applied to a network of ambient objects. Colonies of ants have a very high capacity to adapt themselves to changes in the environment as well as an enormous strength in dealing with situations where one individual fails to carry out a given task. This kind of flexibility would also be very useful for mobile networks of objects which are perpetually developing. Parcels of information that move from a computer to a digital object behave in the same way as ants would do. They move through the network and pass from one knot to the next with the objective of arriving at their final destination as quickly as possible.[[13]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-13)

### Artificial pheromone system**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=3" \o "Edit section: Artificial pheromone system)]**

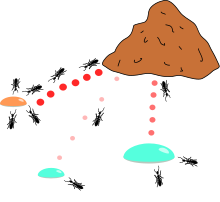
Pheromone-based communication is one of the most effective ways of communication which is widely observed in nature. Pheromone is used by social insects such as bees, ants and termites; both for inter-agent and agent-swarm communications. Due to its feasibility, artificial pheromones have been adopted in multi-robot and swarm robotic systems. Pheromone-based communication was implemented by different means such as chemical [[14]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-14)[[15]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-15)[[16]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-16) or physical (RFID tags,[[17]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-17) light,[[18]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-18)[[19]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-19)[[20]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-20)[[21]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-21) sound[[22]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-22)) ways. However, those implementations were not able to replicate all the aspects of pheromones as seen in nature.

Using projected light was presented in an 2007 IEEE paper by Garnier, Simon, et al. as an experimental setup to study pheromone-based communication with micro autonomous robots.[[23]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-23) Another study presented a system in which pheromones were implemented via a horizontal LCD screen on which the robots moved, with the robots having downward facing light sensors to register the patterns beneath them.[[24]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-24)[[25]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-25)

## **Algorithm and formula[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=4" \o "Edit section: Algorithm and formula)]**

In the ant colony optimization algorithms, an artificial ant is a simple computational agent that searches for good solutions to a given optimization problem. To apply an ant colony algorithm, the optimization problem needs to be converted into the problem of finding the [shortest path](https://en.wikipedia.org/wiki/Shortest_path_problem) on a weighted graph. In the first step of each iteration, each ant stochastically constructs a solution, i.e. the order in which the edges in the graph should be followed. In the second step, the paths found by the different ants are compared. The last step consists of updating the pheromone levels on each edge.

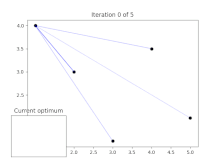
Applications[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=16" \o "Edit section: Applications)]

[](https://en.wikipedia.org/wiki/File:Knapsack_ants.svg)

[Knapsack problem](https://en.wikipedia.org/wiki/Knapsack_problem): The ants prefer the smaller drop of honey over the more abundant, but less nutritious, sugar

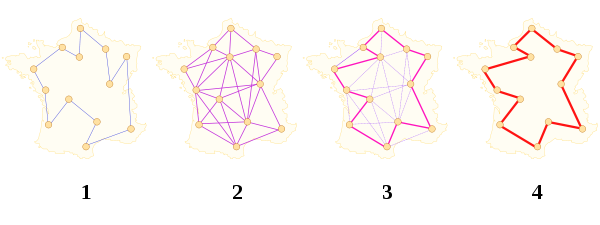
Ant colony optimization algorithms have been applied to many [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization) problems, ranging from quadratic assignment to [protein](https://en.wikipedia.org/wiki/Protein) folding or [routing vehicles](https://en.wikipedia.org/wiki/Vehicle_routing_problem) and a lot of derived methods have been adapted to dynamic problems in real variables, stochastic problems, multi-targets and [parallel](https://en.wikipedia.org/wiki/Parallel_computing) implementations. It has also been used to produce near-optimal solutions to the [travelling salesman problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem). They have an advantage over [simulated annealing](https://en.wikipedia.org/wiki/Simulated_annealing) and [genetic algorithm](https://en.wikipedia.org/wiki/Genetic_algorithm) approaches of similar problems 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](https://en.wikipedia.org/wiki/Network_routing) and urban transportation systems.

The first ACO algorithm was called the ant system[[26]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms" \l "cite_note-Ant_system-26) and it was aimed to solve the travelling salesman problem, in which the goal is to find the shortest round-trip to link a series of cities. The general algorithm is relatively simple and based on a set of ants, each making one of the possible round-trips along the cities. At each stage, the ant chooses to move from one city to another according to some rules:

[](https://en.wikipedia.org/wiki/File:Ant_Colony_Algorihm_applied_to_the_Travelling_Salesman_Problem.gif)

Visualization of the ant colony algorithm applied to the travelling salesman problem. The green lines are the paths chosen by each ant. The blue lines are the paths it may take at each point. When the ant finishes, the pheromone levels are represented in red.

1. It must visit each city exactly once;
2. A distant city has less chance of being chosen (the visibility);
3. The more intense the pheromone trail laid out on an edge between two cities, the greater the probability that that edge will be chosen;
4. Having completed its journey, the ant deposits more pheromones on all edges it traversed, if the journey is short;
5. After each iteration, trails of pheromones evaporate.

[](https://en.wikipedia.org/wiki/File:Aco_TSP.svg)

**Scheduling problem**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=17" \o "Edit section: Scheduling problem)]

* [Sequential ordering problem](https://en.wikipedia.org/w/index.php?title=Sequential_ordering_problem&action=edit&redlink=1) (SOP) [[34]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-34)
* [Job-shop scheduling](https://en.wikipedia.org/wiki/Job-shop_scheduling) problem (JSP)[[35]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-35)
* [Open-shop scheduling](https://en.wikipedia.org/wiki/Open-shop_scheduling) problem (OSP)[[36]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-36)[[37]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-37)
* Permutation flow shop problem (PFSP)[[38]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-38)
* Single machine total tardiness problem (SMTTP)[[39]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-39)
* Single machine total weighted tardiness problem (SMTWTP)[[40]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-40)[[41]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-41)[[42]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-42)
* Resource-constrained project scheduling problem (RCPSP)[[43]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-43)
* Group-shop scheduling problem (GSP)[[44]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-44)
* Single-machine total tardiness problem with sequence dependent setup times (SMTTPDST)[[45]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-45)
* Multistage flowshop scheduling problem (MFSP) with sequence dependent setup/changeover times[[46]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-46)
* Assembly Sequence Planning (ASP) problems[[47]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-47)

**Vehicle routing problem**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=18" \o "Edit section: Vehicle routing problem)]

* Capacitated vehicle routing problem (CVRP)[[48]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-48)[[49]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-49)[[50]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-50)
* Multi-depot vehicle routing problem (MDVRP)[[51]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-51)
* Period vehicle routing problem (PVRP)[[52]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-52)
* Split delivery vehicle routing problem (SDVRP)[[53]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-53)
* Stochastic vehicle routing problem (SVRP)[[54]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-54)
* Vehicle routing problem with pick-up and delivery (VRPPD)[[55]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-55)[[56]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-56)
* Vehicle routing problem with time windows (VRPTW)[[57]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-L.M._Gambardella,_E._Taillard_pp._63-76-57)[[58]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-58)[[59]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-59)[[60]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-60)
* Time dependent vehicle routing problem with time windows (TDVRPTW)[[61]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-61)
* Vehicle routing problem with time windows and multiple service workers (VRPTWMS)

**Assignment problem**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=19" \o "Edit section: Assignment problem)]

* [Quadratic assignment problem](https://en.wikipedia.org/wiki/Quadratic_assignment_problem) (QAP)[[62]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-62)
* [Generalized assignment problem](https://en.wikipedia.org/wiki/Generalized_assignment_problem) (GAP)[[63]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-63)[[64]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-64)
* [Frequency assignment problem](https://en.wikipedia.org/w/index.php?title=Frequency_assignment_problem&action=edit&redlink=1) (FAP)[[65]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-65)
* [Redundancy allocation problem](https://en.wikipedia.org/w/index.php?title=Redundancy_allocation_problem&action=edit&redlink=1) (RAP)[[66]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-66)

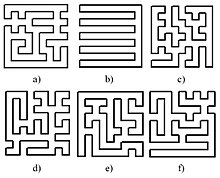
**Set problem**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=20" \o "Edit section: Set problem)]

* [Set cover problem](https://en.wikipedia.org/wiki/Set_cover_problem) (SCP)[[67]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-67)[[68]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-68)
* [Partition problem](https://en.wikipedia.org/wiki/Partition_problem) (SPP)[[69]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-69)
* Weight constrained graph tree partition problem (WCGTPP)[[70]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-70)
* Arc-weighted l-cardinality tree problem (AWlCTP)[[71]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-71)
* Multiple knapsack problem (MKP)[[72]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-72)
* Maximum independent set problem (MIS)[[73]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-73)

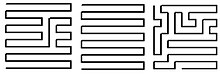
**Device sizing problem in nanoelectronics physical design**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=21" \o "Edit section: Device sizing problem in nanoelectronics physical design)]

* Ant colony optimization (ACO) based optimization of 45 nm CMOS-based sense amplifier circuit could converge to optimal solutions in very minimal time.[[74]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-74)
* Ant colony optimization (ACO) based reversible circuit synthesis could improve efficiency significantly.[[75]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-75)

**Antennas optimization and synthesis**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=22" \o "Edit section: Antennas optimization and synthesis)]

[](https://en.wikipedia.org/wiki/File:ANT_Antenna_1.jpg)

Loopback vibrators 10×10, synthesized by means of ACO algorithm[[76]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms" \l "cite_note-slyusarant1-76)

[](https://en.wikipedia.org/wiki/File:ANT_antenna_2.jpg)

Unloopback vibrators 10×10, synthesized by means of ACO algorithm[[76]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms" \l "cite_note-slyusarant1-76)

To optimize the form of antennas, ant colony algorithms can be used. As example can be considered antennas RFID-tags based on ant colony algorithms (ACO),[[77]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms" \l "cite_note-77) loopback and unloopback vibrators 10×10[[76]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-slyusarant1-76)

**Image processing**[[edit](https://en.wikipedia.org/w/index.php?title=Ant_colony_optimization_algorithms&action=edit&section=23" \o "Edit section: Image processing)]

The ACO algorithm is used in image processing for image edge detection and edge linking.[[78]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-78)[[79]](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms#cite_note-79)

* **Edge detection:**

The graph here is the 2-D image and the ants traverse from one pixel depositing pheromone. The movement of ants from one pixel to another is directed by the local variation of the image's intensity values. This movement causes the highest density of the pheromone to be deposited at the edges.

#### River formation dynamics (Rabanal, Rodríguez & Rubio, 2007)**[[edit](https://en.wikipedia.org/w/index.php?title=List_of_metaphor-based_metaheuristics&action=edit&section=11" \o "Edit section: River formation dynamics (Rabanal, Rodríguez & Rubio, 2007))]**

River formation dynamics is based on imitating how water forms rivers by eroding the ground and depositing sediments (the drops act as the swarm). After drops transform the landscape by increasing/decreasing the altitude of places, solutions are given in the form of paths of decreasing altitudes. Decreasing gradients are constructed, and these gradients are followed by subsequent drops to compose new gradients and reinforce the best ones. This heuristic optimization method was proposed in 2007 by Rabanal et al.[[31]](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics#cite_note-31) The applicability of RFD to other NP-complete problems has been studied,[[32]](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics" \l "cite_note-32) and the algorithm has been applied to fields such as routing[[33]](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics#cite_note-33) and robot navigation.[[34]](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics#cite_note-34) The main applications of RFD can be found at the survey Rabanal et al. (2017).[[35]](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics#cite_note-35)

#### Gravitational search algorithm (Rashedi, Nezamabadi-pour & Saryazdi, 2009)**[[edit](https://en.wikipedia.org/w/index.php?title=List_of_metaphor-based_metaheuristics&action=edit&section=12" \o "Edit section: Gravitational search algorithm (Rashedi, Nezamabadi-pour & Saryazdi, 2009))]**

The gravitational search algorithm is based on the [law of gravity](https://en.wikipedia.org/wiki/Law_of_gravity) and the notion of mass interactions. The GSA algorithm uses the theory of [Newtonian physics](https://en.wikipedia.org/wiki/Newtonian_physics) and its searcher [agents](https://en.wikipedia.org/wiki/Agent_(artificial_intelligence)) are the collection of masses. In GSA, there is an [isolated system](https://en.wikipedia.org/wiki/Isolated_system) of masses. Using the gravitational force, every mass in the system can [see the situation](https://en.wikipedia.org/wiki/Action_at_a_distance) of other masses. The gravitational force is therefore a way of transferring information between different masses.[[36]](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics#cite_note-36) In GSA, agents are considered as objects and their performance is measured by their masses. All these objects attract each other by a [gravity](https://en.wikipedia.org/wiki/Gravity) force, and this force causes movement of all objects towards the objects with heavier masses. Heavier masses correspond to better solutions of the problem. The [position](https://en.wikipedia.org/wiki/Position_(vector)) of the agent corresponds to a solution of the problem, and its mass is determined using a fitness function. By lapse of time, masses are attracted by the heaviest mass, which would ideally present an optimum solution in the search space. The GSA could be considered as a small artificial world of masses obeying the Newtonian laws of gravitation and motion.[[37]](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics#cite_note-37) A multi-objective variant of GSA, called MOGSA, was proposed by Hassanzadeh et al. in 2010.[38]

Application of an Ant Colony Optimization Algorithm for Optimal Operation of Reservoirs: A Comparative Study

of Three Proposed Formulations R. Moeini1 and M.H. Afshar1;

Abstract. This paper presents an application of the Max-Min Ant System for optimal operation of reservoirs using three dierent formulations. Ant colony optimization algorithms are a meta-heuristic approach initially inspired by the observation that ants can and the shortest path between food sources and their nest. The basic algorithm of Ant Colony Optimization is the Ant System. Many other algorithms, such as the Max-Min Ant System, have been introduced to improve the performance of the Ant System. The rst step for solving problems using ant algorithms is to de ne the graph of the problem under consideration. The problem graph is related to the decision variables of problems. In this paper, the problem of optimal operation of reservoirs is formulated using two di  
erent sets of decision variable, i.e. storage volumes and releases. It is also shown that the problem can be formulated in two dierent graph forms when the reservoir storages are taken as the decision variables, while only one graph representation is available when the releases are taken as the decision variables. The advantages and disadvantages of these formulation are discussed when an ant algorithm, such as the Max-Min Ant System, is attempted to solve the underlying problem. The proposed formulations are then used to solve the problem of water supply and the hydropower operation of the \Dez" reservoir. The results are then compared with each other and those of other methods such as the Ant Colony System, Genetic Algorithms, Honey Bee Mating Optimization and the results obtained by Lingo software. The results indicate the ability of the proposed formulation and, in particular, the third formulation to optimally solve reservoir operation problems. Keywords: Ant colony optimization; Max-Min Ant System; Graph; Optimal operation of reservoir; Hydropower reservoir.

Ant Colony Optimization for Optimized Operation Scheduling of Combined Heat and Power Plants <https://link.springer.com/chapter/10.1007/978-3-030-16692-2_7>

A Comparison of Different Many-Objective Optimization Algorithms for Energy System Optimization <https://link.springer.com/chapter/10.1007/978-3-030-16692-2_1>

Application of evolutionary algorithm in performance optimization of embedded network firewall <https://www.sciencedirect.com/science/article/abs/pii/S0141933120301551>

Application of ant colony optimization algorithm in integrated process planning and scheduling <https://link.springer.com/article/10.1007/s00170-015-8145-4>

Application of ant colony optimization algorithm in process planning optimization <https://link.springer.com/article/10.1007/s10845-010-0407-2>

Swarm Intelligence for Resource Management in Internet of Things <https://www.sciencedirect.com/science/article/pii/B9780128182871000188>

Swarm Intelligence for Resource Management in Internet of Things Intelligent Data-Centric Systems 2020, Pages 1-19 <https://www.sciencedirect.com/science/article/pii/B9780128182871000036>

Swarm intelligence

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/Swarm_intelligence#mw-head)[Jump to search](https://en.wikipedia.org/wiki/Swarm_intelligence#searchInput)

**Swarm intelligence** (**SI**) is the [collective behavior](https://en.wikipedia.org/wiki/Collective_behavior) of [decentralized](https://en.wikipedia.org/wiki/Decentralization), [self-organized](https://en.wikipedia.org/wiki/Self-organization) systems, natural or artificial. The concept is employed in work on [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). The expression was introduced by [Gerardo Beni](https://en.wikipedia.org/wiki/Gerardo_Beni) and Jing Wang in 1989, in the context of cellular robotic systems.[[1]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-1)

SI systems consist typically of a population of simple [agents](https://en.wikipedia.org/wiki/Intelligent_agent) or [boids](https://en.wikipedia.org/wiki/Boids" \o "Boids) interacting locally with one another and with their environment.[[2]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-tcds-2) The inspiration often comes from nature, especially biological systems. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such agents lead to the [emergence](https://en.wikipedia.org/wiki/Emergence) of "intelligent" global behavior, unknown to the individual agents.[[3]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-tro-3) Examples of swarm intelligence in natural systems include [ant colonies](https://en.wikipedia.org/wiki/Ant_colony), [bee colonies](https://en.wikipedia.org/wiki/Bee_colonies), bird [flocking](https://en.wikipedia.org/wiki/Flocking_(behavior)), hawks [hunting](https://en.wikipedia.org/wiki/Hunting), animal [herding](https://en.wikipedia.org/wiki/Herding), [bacterial growth](https://en.wikipedia.org/wiki/Bacteria#Growth_and_reproduction), fish [schooling](https://en.wikipedia.org/wiki/Shoaling_and_schooling) and [microbial intelligence](https://en.wikipedia.org/wiki/Microbial_intelligence).

The application of swarm principles to [robots](https://en.wikipedia.org/wiki/Robot) is called [*swarm robotics*](https://en.wikipedia.org/wiki/Swarm_robotics) while *swarm intelligence* refers to the more general set of algorithms. *Swarm prediction* has been used in the context of forecasting problems. Similar approaches to those proposed for swarm robotics are considered for [genetically modified organisms](https://en.wikipedia.org/wiki/Genetically_modified_organisms) in synthetic collective intelligence.[[4]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-4)

Models of swarm behavior[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=1" \o "Edit section: Models of swarm behavior)]

*See also:*[*Swarm behaviour*](https://en.wikipedia.org/wiki/Swarm_behaviour)

**Boids (Reynolds 1987)**[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=2" \o "Edit section: Boids (Reynolds 1987))]

*Main article: [Boids](https://en.wikipedia.org/wiki/Boids" \o "Boids)*

Boids is an [artificial life](https://en.wikipedia.org/wiki/Artificial_life) program, developed by [Craig Reynolds](https://en.wikipedia.org/wiki/Craig_Reynolds_(computer_graphics)) in 1986, which simulates the [flocking](https://en.wikipedia.org/wiki/Flocking_(behavior)) behaviour of birds. His paper on this topic was published in 1987 in the proceedings of the [ACM](https://en.wikipedia.org/wiki/Association_for_Computing_Machinery) [SIGGRAPH](https://en.wikipedia.org/wiki/SIGGRAPH) conference.[[5]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-reynolds-5) The name "boid" corresponds to a shortened version of "bird-oid object", which refers to a bird-like object.[[6]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-6)

As with most artificial life simulations, Boids is an example of [emergent](https://en.wikipedia.org/wiki/Emergence) behavior; that is, the complexity of Boids arises from the interaction of individual agents (the boids, in this case) adhering to a set of simple rules. The rules applied in the simplest Boids world are as follows:

* **separation**: [steer](https://en.wiktionary.org/wiki/steer#Verb) to avoid crowding local flockmates
* **alignment**: steer towards the average heading of local flockmates
* **cohesion**: steer to move toward the average position (center of mass) of local flockmates

More complex rules can be added, such as obstacle avoidance and goal seeking.

**Self-propelled particles (Vicsek *et al*. 1995)**[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=3" \o "Edit section: Self-propelled particles (Vicsek et al. 1995))]

*Main article:*[*Self-propelled particles*](https://en.wikipedia.org/wiki/Self-propelled_particles)

Self-propelled particles (SPP), also referred to as the *[Vicsek model](https://en.wikipedia.org/wiki/Vicsek_model" \o "Vicsek model)*, was introduced in 1995 by [Vicsek](https://en.wikipedia.org/wiki/Tam%C3%A1s_Vicsek" \o "Tamás Vicsek) *et al.*[[7]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-Vicsek1995-7) as a special case of the [boids](https://en.wikipedia.org/wiki/Boids" \o "Boids) model introduced in 1986 by [Reynolds](https://en.wikipedia.org/wiki/Craig_Reynolds_(computer_graphics)).[[5]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-reynolds-5) A swarm is modelled in SPP by a collection of particles that move with a constant speed but respond to a random perturbation by adopting at each time increment the average direction of motion of the other particles in their local neighbourhood.[[8]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-8) SPP models predict that swarming animals share certain properties at the group level, regardless of the type of animals in the swarm.[[9]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-Buhl_et_al-9) Swarming systems give rise to [emergent behaviours](https://en.wikipedia.org/wiki/Emergent_behaviour) which occur at many different scales, some of which are turning out to be both universal and robust. It has become a challenge in theoretical physics to find minimal statistical models that capture these behaviours.[[10]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-10)[[11]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-Bertin_et_al-11)[[12]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-Li_et_al-12)

Metaheuristics[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=4" \o "Edit section: Metaheuristics)]

*See also:*[*List of metaphor-based metaheuristics*](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics)

[Evolutionary algorithms](https://en.wikipedia.org/wiki/Evolutionary_algorithm) (EA), [particle swarm optimization](https://en.wikipedia.org/wiki/Particle_swarm_optimization) (PSO), [differential evolution](https://en.wikipedia.org/wiki/Differential_evolution) (DE), [ant colony optimization](https://en.wikipedia.org/wiki/Ant_colony_optimization) (ACO) and their variants dominate the field of nature-inspired [metaheuristics](https://en.wikipedia.org/wiki/Metaheuristic).[[13]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-13) This list includes algorithms published up to circa the year 2000. A large number of more recent metaphor-inspired metaheuristics have started to [attract criticism in the research community](https://en.wikipedia.org/wiki/List_of_metaphor-inspired_metaheuristics#Criticism) for hiding their lack of novelty behind an elaborate metaphor. For algorithms published since that time, see [List of metaphor-based metaheuristics](https://en.wikipedia.org/wiki/List_of_metaphor-based_metaheuristics).

[Metaheuristics](https://en.wikipedia.org/wiki/Metaheuristic) lack a confidence in a solution.[[14]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-Silberholz_2019_581%E2%80%93604-14) When appropriate parameters are determined, and when sufficient convergence stage is achieved, they often find a solution that is optimal, or near close to optimum – nevertheless, if one does not know optimal solution in advance, a quality of a solution is not known.[[14]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-Silberholz_2019_581%E2%80%93604-14) In spite of this obvious drawback it has been shown that these types of [algorithms](https://en.wikipedia.org/wiki/Algorithm) work well in practice, and have been extensively researched, and developed.[[15]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-15)[[16]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-16)[[17]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-17)[[18]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-18)[[19]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-19) On the other hand, it is possible to avoid this drawback by calculating solution quality for a special case where such calculation is possible, and after such run it is known that every solution that is at least as good as the solution a special case had, has at least a solution confidence a special case had. One such instance is [Ant](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms) inspired [Monte Carlo algorithm](https://en.wikipedia.org/wiki/Monte_Carlo_algorithm) for [Minimum Feedback Arc Set](https://en.wikipedia.org/wiki/Minimum_feedback_arc_set) where this has been achieved probabilistically via hybridization of [Monte Carlo algorithm](https://en.wikipedia.org/wiki/Monte_Carlo_algorithm) with [Ant Colony Optimization](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms) technique.[[20]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-20)

**Stochastic diffusion search (Bishop 1989)**[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=5" \o "Edit section: Stochastic diffusion search (Bishop 1989))]

*Main article:*[*Stochastic diffusion search*](https://en.wikipedia.org/wiki/Stochastic_diffusion_search)

First published in 1989 Stochastic diffusion search (SDS)[[21]](https://en.wikipedia.org/wiki/Swarm_intelligence" \l "cite_note-21)[[22]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-22) was the first Swarm Intelligence metaheuristic. SDS is an agent-based [probabilistic](https://en.wikipedia.org/wiki/Probabilistic_algorithm) global search and optimization technique best suited to problems where the objective function can be decomposed into multiple independent partial-functions. Each agent maintains a hypothesis that is iteratively tested by evaluating a randomly selected partial objective function parameterised by the agent's current hypothesis. In the standard version of SDS such partial function evaluations are binary, resulting in each agent becoming active or inactive. Information on hypotheses is diffused across the population via inter-agent communication. Unlike the [stigmergic](https://en.wikipedia.org/wiki/Stigmergy" \o "Stigmergy) communication used in ACO, in SDS agents communicate [hypotheses](https://en.wikipedia.org/wiki/Hypothesis) via a one-to-one communication strategy analogous to the [tandem running](https://en.wikipedia.org/wiki/Tandem_running) procedure observed in [Leptothorax acervorum](https://en.wikipedia.org/wiki/Leptothorax_acervorum" \o "Leptothorax acervorum).[[23]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-23) A positive feedback mechanism ensures that, over time, a population of agents stabilise around the global-best solution. SDS is both an efficient and robust global search and optimisation algorithm, which has been extensively mathematically described.[[24]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-24)[[25]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-25)[[26]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-26) Recent work has involved merging the global search properties of SDS with other swarm intelligence algorithms.[[27]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-27)[[28]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-28)

**Ant colony optimization (Dorigo 1992)**[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=6" \o "Edit section: Ant colony optimization (Dorigo 1992))]

*Main article:*[*Ant colony optimization*](https://en.wikipedia.org/wiki/Ant_colony_optimization)

Ant colony optimization (ACO), introduced by Dorigo in his doctoral dissertation, is a class of [optimization](https://en.wikipedia.org/wiki/Optimization_(mathematics)) [algorithms](https://en.wikipedia.org/wiki/Algorithm) modeled on the actions of an [ant colony](https://en.wikipedia.org/wiki/Ant_colony). ACO is a [probabilistic technique](https://en.wikipedia.org/wiki/Probabilistic_algorithm) useful in problems that deal with finding better paths through graphs. Artificial 'ants'—simulation agents—locate optimal solutions by moving through a [parameter space](https://en.wikipedia.org/wiki/Parameter_space) representing all possible solutions. Natural ants lay down [pheromones](https://en.wikipedia.org/wiki/Pheromone) directing each other to resources while exploring their environment. The simulated 'ants' similarly record their positions and the quality of their solutions, so that in later simulation iterations more ants locate for better solutions.[[29]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-29)

**Particle swarm optimization (Kennedy, Eberhart & Shi 1995)**[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=7" \o "Edit section: Particle swarm optimization (Kennedy, Eberhart & Shi 1995))]

*Main article:*[*Particle swarm optimization*](https://en.wikipedia.org/wiki/Particle_swarm_optimization)

Particle swarm optimization (PSO) is a [global optimization](https://en.wikipedia.org/wiki/Global_optimization) algorithm for dealing with problems in which a best solution can be represented as a point or surface in an n-dimensional space. Hypotheses are plotted in this space and seeded with an initial [velocity](https://en.wikipedia.org/wiki/Velocity), as well as a communication channel between the particles.[[30]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-30)[[31]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-31) Particles then move through the solution space, and are evaluated according to some [fitness](https://en.wikipedia.org/wiki/Fitness_(biology)) criterion after each timestep. Over time, particles are accelerated towards those particles within their communication grouping which have better fitness values. The main advantage of such an approach over other global minimization strategies such as [simulated annealing](https://en.wikipedia.org/wiki/Simulated_annealing) is that the large number of members that make up the particle swarm make the technique impressively resilient to the problem of [local minima](https://en.wikipedia.org/wiki/Local_minima).

**Artificial Swarm Intelligence (2015)**[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=8" \o "Edit section: Artificial Swarm Intelligence (2015))]

Artificial Swarm Intelligence (ASI) is method of amplifying the collective intelligence of networked human groups using control algorithms modeled after natural swarms. Sometimes referred to as Human Swarming or Swarm AI, the technology connects groups of human participants into real-time systems that deliberate and converge on solutions as dynamic swarms when simultaneously presented with a question[[32]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-32)[[33]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-:0-33)[[34]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-34) ASI has been used for a wide range of applications, from enabling business teams to generate highly accurate financial forecasts[[35]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-35) to enabling sports fans to outperform Vegas betting markets.[[36]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-:2-36) ASI has also been used to enable groups of doctors to generate diagnoses with significantly higher accuracy than traditional methods.[[37]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-:4-37)[[38]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-:3-38) ASI has been used by the [Food and Agriculture Organization (FAO)](https://en.wikipedia.org/wiki/Food_and_Agriculture_Organization) of the [United Nations](https://en.wikipedia.org/wiki/United_Nations) to help forecast famines in hotspots around the world.[[39]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-39)[[*better source needed*](https://en.wikipedia.org/wiki/Wikipedia:NOTRS)]

Applications[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=9" \o "Edit section: Applications)]

Swarm Intelligence-based techniques can be used in a number of applications. The U.S. military is investigating swarm techniques for controlling unmanned vehicles. The [European Space Agency](https://en.wikipedia.org/wiki/European_Space_Agency) is thinking about an orbital swarm for self-assembly and interferometry. [NASA](https://en.wikipedia.org/wiki/NASA) is investigating the use of swarm technology for planetary mapping. A 1992 paper by [M. Anthony Lewis](https://en.wikipedia.org/wiki/M._Anthony_Lewis_(roboticist)) and [George A. Bekey](https://en.wikipedia.org/wiki/George_A._Bekey) discusses the possibility of using swarm intelligence to control nanobots within the body for the purpose of killing cancer tumors.[[40]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-40) Conversely al-Rifaie and Aber have used [stochastic diffusion search](https://en.wikipedia.org/wiki/Stochastic_diffusion_search) to help locate tumours.[[41]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-41)[[42]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-42) Swarm intelligence has also been applied for [data mining](https://en.wikipedia.org/wiki/Data_mining)[[43]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-43) and [cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis).[[44]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-44) Ant based models are further subject of modern management theory.[[45]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-45)

**Ant-based routing**[[edit](https://en.wikipedia.org/w/index.php?title=Swarm_intelligence&action=edit&section=10" \o "Edit section: Ant-based routing)]

The use of swarm intelligence in [telecommunication networks](https://en.wikipedia.org/wiki/Telecommunications_network) has also been researched, in the form of [ant-based routing](https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms). This was pioneered separately by Dorigo et al. and [Hewlett Packard](https://en.wikipedia.org/wiki/Hewlett_Packard) in the mid-1990s, with a number of variants existing. Basically, this uses a [probabilistic](https://en.wikipedia.org/wiki/Probabilistic_algorithm) routing table rewarding/reinforcing the route successfully traversed by each "ant" (a small control packet) which flood the network. Reinforcement of the route in the forwards, reverse direction and both simultaneously have been researched: backwards reinforcement requires a symmetric network and couples the two directions together; forwards reinforcement rewards a route before the outcome is known (but then one would pay for the cinema before one knows how good the film is). As the system behaves stochastically and is therefore lacking repeatability, there are large hurdles to commercial deployment. Mobile media and new technologies have the potential to change the threshold for collective action due to swarm intelligence (Rheingold: 2002, P175).

The location of transmission infrastructure for wireless communication networks is an important engineering problem involving competing objectives. A minimal selection of locations (or sites) are required subject to providing adequate area coverage for users. A very different-ant inspired swarm intelligence algorithm, stochastic diffusion search (SDS), has been successfully used to provide a general model for this problem, related to circle packing and set covering. It has been shown that the SDS can be applied to identify suitable solutions even for large problem instances.[[46]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-46)

Airlines have also used ant-based routing in assigning aircraft arrivals to airport gates. At [Southwest Airlines](https://en.wikipedia.org/wiki/Southwest_Airlines) a software program uses swarm theory, or swarm intelligence—the idea that a colony of ants works better than one alone. Each pilot acts like an ant searching for the best airport gate. "The pilot learns from his experience what's the best for him, and it turns out that that's the best solution for the airline," [Douglas A. Lawson](https://en.wikipedia.org/wiki/Douglas_A._Lawson) explains. As a result, the "colony" of pilots always go to gates they can arrive at and depart from quickly. The program can even alert a pilot of plane back-ups before they happen. "We can anticipate that it's going to happen, so we'll have a gate available," Lawson says.[[47]](https://en.wikipedia.org/wiki/Swarm_intelligence#cite_note-47)

**Automated planning and scheduling**, sometimes denoted as simply **AI planning**,[[1]](https://en.wikipedia.org/wiki/Automated_planning_and_scheduling#cite_note-1) is a branch of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) that concerns the realization of [strategies](https://en.wikipedia.org/wiki/Strategy) or action sequences, typically for execution by [intelligent agents](https://en.wikipedia.org/wiki/Intelligent_agent), [autonomous robots](https://en.wikipedia.org/wiki/Autonomous_robot) and [unmanned vehicles](https://en.wikipedia.org/wiki/Unmanned_vehicle). Unlike classical [control](https://en.wikipedia.org/wiki/Control_system) and [classification](https://en.wikipedia.org/wiki/Statistical_classification) problems, the solutions are complex and must be discovered and optimized in multidimensional space. Planning is also related to [decision theory](https://en.wikipedia.org/wiki/Decision_theory).

In known environments with available models, planning can be done offline. Solutions can be found and evaluated prior to execution. In dynamically unknown environments, the [strategy](https://en.wikipedia.org/wiki/Strategy) often needs to be revised online. Models and policies must be adapted. Solutions usually resort to iterative [trial and error](https://en.wikipedia.org/wiki/Trial_and_error) processes commonly seen in [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). These include [dynamic programming](https://en.wikipedia.org/wiki/Dynamic_programming), [reinforcement learning](https://en.wikipedia.org/wiki/Reinforcement_learning) and [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization). Languages used to describe planning and scheduling are often called [action languages](https://en.wikipedia.org/wiki/Action_language).

Design Automation usually refers to [electronic design automation](https://en.wikipedia.org/wiki/Electronic_design_automation), or [Design Automation](https://en.wikipedia.org/wiki/Design_Automation) which is a [Product Configurator](https://en.wikipedia.org/wiki/Product_Configurator). Extending [Computer-Aided Design](https://en.wikipedia.org/wiki/Computer-Aided_Design) (CAD), automated design and **Computer-Automated Design (CAutoD)**[[1]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-IBM-1)[[2]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-3) are more concerned with a broader range of applications, such as [automotive engineering](https://en.wikipedia.org/wiki/Automotive_engineering), [civil engineering](https://en.wikipedia.org/wiki/Civil_engineering),[[4]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-4)[[5]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-5)[[6]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-6)[[7]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-7) [composite material](https://en.wikipedia.org/wiki/Composite_material) design, [control engineering](https://en.wikipedia.org/wiki/Control_engineering),[[8]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-8) dynamic [system identification](https://en.wikipedia.org/wiki/System_identification) and optimization,[[9]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-9) [financial](https://en.wikipedia.org/wiki/Financial) systems, industrial equipment, [mechatronic](https://en.wikipedia.org/wiki/Mechatronic) systems, [steel construction](https://en.wikipedia.org/wiki/Steel_construction),[[10]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-10) structural [optimisation](https://en.wikipedia.org/wiki/Optimization_(mathematics)" \o "Optimization (mathematics)),[[11]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-11) and the invention of novel systems.[[12]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-12)

The concept of CAutoD perhaps first appeared in 1963, in the IBM Journal of Research and Development,[[1]](https://en.wikipedia.org/wiki/Computer-automated_design" \l "cite_note-IBM-1) where a computer program was written.

1. to search for logic circuits having certain constraints on hardware design
2. to evaluate these logics in terms of their discriminating ability over samples of the character set they are expected to recognize.

More recently, traditional CAD simulation is seen to be transformed to CAutoD by biologically-inspired [machine learning](https://en.wikipedia.org/wiki/Machine_learning),[[13]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-13) including heuristic [search techniques](https://en.wikipedia.org/wiki/Search_algorithm) such as [evolutionary computation](https://en.wikipedia.org/wiki/Evolutionary_computation),[[14]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-14)[[15]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-15) and [swarm intelligence](https://en.wikipedia.org/wiki/Swarm_intelligence) algorithms.[[16]](https://en.wikipedia.org/wiki/Computer-automated_design#cite_note-16)

The **vehicle routing problem** (**VRP**) is a [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization) and [integer programming](https://en.wikipedia.org/wiki/Integer_programming) problem which asks "What is the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers?". It generalises the well-known [travelling salesman problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem) (TSP). It first appeared in a paper by [George Dantzig](https://en.wikipedia.org/wiki/George_Dantzig) and John Ramser in 1959,[[1]](https://en.wikipedia.org/wiki/Vehicle_routing_problem" \l "cite_note-DantzigRamser1959-1) in which the first algorithmic approach was written and was applied to petrol deliveries. Often, the context is that of delivering goods located at a central depot to customers who have placed orders for such goods. The objective of the VRP is to minimize the total route cost. In 1964, Clarke and Wright improved on Dantzig and Ramser's approach using an effective [greedy algorithm](https://en.wikipedia.org/wiki/Greedy_algorithm) called the savings algorithm.

Determining the optimal solution to VRP is [NP-hard](https://en.wikipedia.org/wiki/NP-hard),[[2]](https://en.wikipedia.org/wiki/Vehicle_routing_problem#cite_note-toth-2) so the size of problems that can be solved, optimally, using [mathematical programming](https://en.wikipedia.org/wiki/Mathematical_programming) or [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization) may be limited. Therefore, commercial solvers tend to use heuristics due to the size and frequency of real world VRPs they need to solve.

VRP has many direct applications in industry. Vendors of VRP routing tools often claim that they can offer cost savings of 5%–30%.[[3]](https://en.wikipedia.org/wiki/Vehicle_routing_problem#cite_note-Springer_Verlag-3)

The **vehicle rescheduling problem (VRSP)** is a [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization) and [integer programming](https://en.wikipedia.org/wiki/Integer_programming) problem seeking to service customers on a trip after change of schedule such as vehicle break down or major delay. Proposed by Li, Mirchandani and Borenstein in 2007,[[1]](https://en.wikipedia.org/wiki/Vehicle_rescheduling_problem" \l "cite_note-LiMirchandani2007-1) the VRSP is an important problem in the fields of transportation and logistics.

Determining the optimal solution is an [NP-complete](https://en.wikipedia.org/wiki/NP-hard) problem in [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization), so in practice heuristic and deterministic methods are used to find acceptably good solutions for the VRSP.

The **travelling salesman problem** (also called the **travelling salesperson problem**or **TSP**) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" It is an [NP-hard](https://en.wikipedia.org/wiki/NP-hardness) problem in [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization), important in [theoretical computer science](https://en.wikipedia.org/wiki/Theoretical_computer_science) and [operations research](https://en.wikipedia.org/wiki/Operations_research).

The [travelling purchaser problem](https://en.wikipedia.org/wiki/Traveling_purchaser_problem) and the [vehicle routing problem](https://en.wikipedia.org/wiki/Vehicle_routing_problem) are both generalizations of TSP.

In the [theory of computational complexity](https://en.wikipedia.org/wiki/Computational_complexity_theory), the decision version of the TSP (where given a length *L*, the task is to decide whether the graph has a tour of at most *L*) belongs to the class of [NP-complete](https://en.wikipedia.org/wiki/NP-completeness) problems. Thus, it is possible that the [worst-case](https://en.wikipedia.org/wiki/Best,_worst_and_average_case) [running time](https://en.wikipedia.org/wiki/Time_complexity) for any algorithm for the TSP increases [superpolynomially](https://en.wikipedia.org/wiki/Time_complexity" \l "Superpolynomial_time" \o "Time complexity) (but no more than [exponentially](https://en.wikipedia.org/wiki/Exponential_time_hypothesis)) with the number of cities.

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a [benchmark](https://en.wikipedia.org/wiki/Benchmark_(computing)) for many optimization methods. Even though the problem is computationally difficult, many [heuristics](https://en.wikipedia.org/wiki/Heuristic) and [exact algorithms](https://en.wikipedia.org/wiki/Exact_algorithm) are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%.[[1]](https://en.wikipedia.org/wiki/Travelling_salesman_problem#cite_note-1)

The TSP has several applications even in its purest formulation, such as [planning](https://en.wikipedia.org/wiki/Planning), [logistics](https://en.wikipedia.org/wiki/Logistics), and the manufacture of [microchips](https://en.wikipedia.org/wiki/Integrated_circuit). Slightly modified, it appears as a sub-problem in many areas, such as [DNA sequencing](https://en.wikipedia.org/wiki/DNA_sequencing). In these applications, the concept *city* represents, for example, customers, soldering points, or DNA fragments, and the concept *distance* represents travelling times or cost, or a [similarity measure](https://en.wikipedia.org/wiki/Similarity_measure) between DNA fragments. The TSP also appears in astronomy, as astronomers observing many sources will want to minimize the time spent moving the telescope between the sources; in such problems, the TSP can be embedded inside an [optimal control problem](https://en.wikipedia.org/wiki/Optimal_control). In many applications, additional constraints such as limited resources or time windows may be imposed.

The **traveling purchaser problem** (**TPP**) is an [NP-hard](https://en.wikipedia.org/wiki/NP-hard) problem studied in [theoretical computer science](https://en.wikipedia.org/wiki/Theoretical_computer_science). Given a list of marketplaces, the cost of travelling between different marketplaces, and a list of available goods together with the price of each such good at each marketplace, the task is to find, for a given list of articles, the route with the minimum combined cost of purchases and traveling. The [traveling salesman problem](https://en.wikipedia.org/wiki/Traveling_salesman_problem) (TSP) is a [special case](https://en.wikipedia.org/wiki/Special_case) of this problem.

In [computational science](https://en.wikipedia.org/wiki/Computational_science), **particle swarm optimization** (**PSO**)[[1]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-bonyadi16survey-1) is a computational method that [optimizes](https://en.wikipedia.org/wiki/Mathematical_optimization) a problem by [iteratively](https://en.wikipedia.org/wiki/Iterative_method) trying to improve a [candidate solution](https://en.wikipedia.org/wiki/Candidate_solution) with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed [particles](https://en.wikipedia.org/wiki/Point_particle), and moving these particles around in the [search-space](https://en.wikipedia.org/wiki/Optimization_(mathematics)#Concepts_and_notation) according to simple [mathematical formula](https://en.wikipedia.org/wiki/Formula) over the particle's [position](https://en.wikipedia.org/wiki/Position_(vector)) and [velocity](https://en.wikipedia.org/wiki/Velocity). Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is originally attributed to [Kennedy](https://en.wikipedia.org/wiki/James_Kennedy_(social_psychologist)), [Eberhart](https://en.wikipedia.org/wiki/Russell_C._Eberhart" \o "Russell C. Eberhart) and Shi[[2]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-kennedy95particle-2)[[3]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-shi98modified-3) and was first intended for [simulating](https://en.wikipedia.org/wiki/Computer_simulation) [social behaviour](https://en.wikipedia.org/wiki/Social_behaviour),[[4]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-kennedy97particle-4) as a stylized representation of the movement of organisms in a bird [flock](https://en.wikipedia.org/wiki/Flocking_(behavior)) or [fish school](https://en.wikipedia.org/wiki/Fish_school). The algorithm was simplified and it was observed to be performing optimization. The book by Kennedy and Eberhart[[5]](https://en.wikipedia.org/wiki/Particle_swarm_optimization" \l "cite_note-kennedy01swarm-5) describes many philosophical aspects of PSO and [swarm intelligence](https://en.wikipedia.org/wiki/Swarm_intelligence). An extensive survey of PSO applications is made by [Poli](https://en.wikipedia.org/wiki/Riccardo_Poli" \o "Riccardo Poli).[[6]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-poli07analysis-6)[[7]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-poli08analysis-7) Recently, a comprehensive review on theoretical and experimental works on PSO has been published by Bonyadi and Michalewicz.[[1]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-bonyadi16survey-1)

PSO is a [metaheuristic](https://en.wikipedia.org/wiki/Metaheuristic) as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. Also, PSO does not use the [gradient](https://en.wikipedia.org/wiki/Gradient) of the problem being optimized, which means PSO does not require that the optimization problem be [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) as is required by classic optimization methods such as [gradient descent](https://en.wikipedia.org/wiki/Gradient_descent) and [quasi-newton methods](https://en.wikipedia.org/wiki/Quasi-newton_methods). However, metaheuristics such as PSO do not guarantee an optimal solution is ever found.

PSO can be related to [molecular dynamics](https://en.wikipedia.org/wiki/Molecular_dynamics).[[8]](https://en.wikipedia.org/wiki/Particle_swarm_optimization#cite_note-8)