
# D4.4 – INITIAL SWARM MODELING LIBRARY

| | |
|---|---|
| Deliverable ID | **D4.4** |
| Deliverable Title | **Initial swarm modeling library** |
| Work Package | **WP4 – Models and algorithms for CPS Swarms** |
| Dissemination Level | **PUBLIC** |
| Version | **1.0** |
| Date | **24/10/2017** |
| Status | **Final** |
| Lead Editor | **Melanie Schranz (LAKE)** |
| Main Contributors | **Etienne Brosse (SOFTEAM), Alessandra Bagnato (SOFTEAM), Wilfried Elmenreich (UNI-KLU)** |

**Published by the CPSwarm Consortium**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731946.

## Document History

| Version | Date | Author(s) | Description |
|---|---|---|---|
| 0.1 | 2017-06-12 | Melanie Schranz (LAKE) | First draft |
| 0.2 | 2017-06-25 | Melanie Schranz (LAKE) | Filled input to TOC |
| 0.3 | 2017-10-03 | Melanie Schranz (LAKE) | Integrate feedback from UNI-KLU |
| 0.4 | 2017-10-05 | Melanie Schranz (LAKE) | Minor revisions focusing the topic |
| 0.5 | 2017-10-12 | Melanie Schranz (LAKE) | Integrate models from SOFTEAM |
| 0.6 | 2017-10-16 | Etienne Brosse (SOFTEAM) | Feedback from SOFTEAM |
| 0.7 | 2017-10-16 | Melanie Schranz (LAKE) | Finalize deliverable for internal review |
| 1.0 | 2017-10-24 | Melanie Schranz (LAKE) | Integrate minor improvements from TTECH and finalize the document |

## Internal Review History

| Review Date | Reviewer | Summary of Comments |
|---|---|---|
| 2017-10-23 | Edin Arnautovic (TTTech) | Accepted with minor comments |
| 2017-10-23 | Omar Morando (DGSKY) | Accepted with minor comments |

# 1 Executive summary

This deliverable, namely "D4.4 Initial swarm modeling library", is a deliverable of the CPSwarm project, funded by the European Comission's Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and innovation program (H2020), reporting the results of the activities carried out by WP4 – Models and algorithms for CPS Swarms. The main objective of the CPSwarm project is to develop a workbench that aims to fully design, develop, validate and deploy engineered swarm solutions. More specifically, the project focuses on modeling of swarms of CPSs, implementing and optimizing the corresponding swarm intelligence algorithms, driven by WP4.

Deliverable D4.4 is a software deliverable and represents a summary of the research work on swarm modeling and swarm intelligence algorithms achieved from M5 to M10 in WP4. Thus, the outcome of the research work on both topics should be seen as an initial step to structure the corresponding models and algorithms utilizing the CPSwarm vision in terms of provided pseudo codes and approaches for the implementation. The models and the algorithms are expected to be extended in their scope.

The most important outcome is the following: In literature, there is no possibility to model swarm algorithms with a common modelling approach for swarm intelligence. Thus, we aim at creating a common swarm modeling approach out of existing swarm modeling concepts. Beside swarm models, also swarm intelligence algorithms are considered.

Furthermore, this deliverable formulates the foundation for a concrete implementation and a proposal for a common swarm modelling approach for "D4.5 – Updated Swarm Modelling Library".

**Table of Contents**

# 2 Introduction

D4.4 – "Initial swarm modeling library" is a public document defining publicly available swarm models and swarm algorithms found till M10 in the CPSwarm project.

LAKE, as deliverable leader, initially drafted the document, which has subsequently been enriched with content, especially from LAKE, SOFTEAM and UNI-KLU.

## 2.1 Scope

This deliverable mainly reports on the outcome of task T4.3, in which we will develop models for swarm behavior. This is the first deliverable in a series of three (D4.4, D4.5, and D4.6), which addresses this topic. Therefore, first of all, D4.4 provides an analysis of the state-of-the-art swarm models and swarm intelligence algorithms. For this purpose, research publications in form of journals, conference papers, technical reports, books, and book chapters are consulted. Especially the swarm intelligence algorithms have been processed to be directly implemented to the library of swarm intelligence algorithms. Furthermore, we suggest a common technique to model these algorithms in two ways: standard (as they are) and customizable. Especially the latter represents a big challenge and need in the swarm modelling community, as often real-world problems cannot be solved by solely applying a specific predefined algorithm.
The presented swarm algorithms refer to CPS2CPS interaction. Another big point are recipes for Human2Swarm interaction. This topic will be the focus of D4.5.

## 2.2 Document organization

The remainder of this deliverable is organized as follows:
Section 3 describes the motivation for extracting swarm intelligence algorithms from nature and applying them to complex (systems of) cyber-physical systems. Section 4 presents the process how swarm intelligence models are extracted from nature and gives examples on selected swarm intelligence models. Section 5 provides an overview on swarm intelligence algorithms. Section 6 draws an initial approach for a common modelling approach for swarm intelligence algorithms. Finally, Section 7 concludes the deliverable and provides an outline on future topics.

## 2.3 Related documents

| ID | Title | Reference | Version | Date |
|---|---|---|---|---|
| D3.1 | Initial System Architecture Analysis & Design Specification | D3.1 | 1.0 | 31/06/2017 |
| D4.1 | Initial CPS Modeling Library | D4.1 | 1.0 | 31/09/2017 |
| D5.2 | Initial CPSwarm Modelling Tool | D5.2 | 1.0 | 31/09/2017 |

# 3 Motivation: Swarm Intelligence in Nature

It is a typical trend in today's technologies: computers are embedded in everyday objects, networked, and equipped with multiple sensors to link the real world with the virtual one – such entities are called cyber-physical systems (CPS). Examples for the applications of CPSs are of different disciplines, collectively referred to IoT, smart mobility, smart grids, or smart houses. CPSs can be described as strongly interconnected hardware and software components. The usage of CPSs and systems of CPS comes with many challenges, including increased dynamics, connectivity, and complexity calling for features like adaptability, scalability, robustness, self- configuration, self-healing, etc. As a system, they are even more complex, hard to control and program. In the project CPSwarm we focus on swarms of CPSs, e.g., swarms of drones, or of ground rovers, and heterogeneous ones.

To handle the complex characteristics of swarms of CPSs, natural systems are taken into consideration that have evolved over millennia to master NP-hard problems (Green, Aleti, & Garcia, 2017). These range from cellular automata over neural networks to swarm intelligence. CPSs can be considered as a different kind of organism: they follow specific behavior that can adapt to changes in the environment including (Hamann & Schmickl, 2012):

- Pursuing a specific goal
- Aggregating or dispersing in the environment
- Communicating
    - direct
    - indirect
- Memorizing
    - states (local, environmental, etc.)
    - morphologies (e.g., size, weight, sensors)

Thus, we can summarize following typical properties characterizing a swarm:

- Individuals with basic capabilities following simple rules
- No central control
    - Decentralized and hence robust
- Emergent
    - Performs complex functions
- Self-organizing

including following main advantages:

- Adaptability
- Robustness
- Scalability.

Moreover, swarm intelligence is not a subfield of artificial intelligence. In (Hamann & Schmickl, 2012) they argue that this is not possible, as *"methods of swarm intelligence are not based on sophisticated and complex engineered reasoning architectures. Instead, swarm intelligence exploits the emergent properties of self-organizing interaction networks operated by computationally inexpensive, reactive and individually often non-cognitive masses of agents. In swarm intelligence, cognition arises, if at all, only on the collective level, thus potentially giving us a new approach to understand our own human cognition."*

## 4 Swarm Intelligence Models

In literature, swarm intelligence models are described as the process of adopting models found in nature in swarm behavior of, e.g., insects, bacteria, or fish (Chee Peng & Dehuri, 2009). Swarm intelligence models are computational models to undertake distributed (optimization) problems in a swarm of, e.g., CPSs. The state-of-the art process follows the steps described in Fig. 1. Nature inspired and still inspires engineers to design similar systems and apply similar algorithms to solve complex real world problems in different domains.

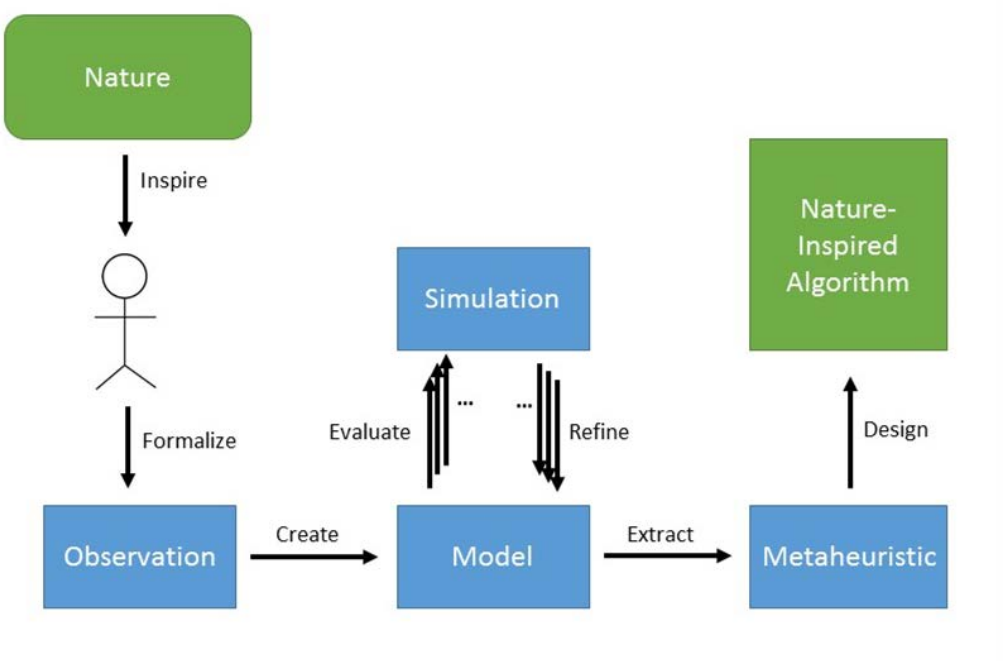



**Fig. 1 Process of designing a swarm intelligence model and the corresponding algorithm (adapted from (Ahmed & Glasgow, 2012)).**

A closer look is taken on the actions, and an analysis of observations enables the creation of a model. The simulation gives then an assessment of how well the intended result can be achieved with a given behavior. This assessment is usually given through a fitness value. Finally, the algorithm is extracted to design a nature or bio-inspired swarm intelligence algorithm.

In all swarm intelligence models, the swarm is made of individual, simple agents. Through direct or indirect communication, they cooperate without a central control. Only through their interactions a collective behavior emerges, which can solve complex tasks. In summary, swarm intelligence models have following characteristics:

- Emergent behavior arises from simple interactions among individuals in a swarm
- Individuals act according to simple and local behavior
- Organized behavior emerges automatically
- There is no central control.

Five basic principles set the basis for swarm intelligence models (Chee Peng & Dehuri, 2009):

1) *Proximity*: ability to perform simple computation of time and space, respond to environmental stimuli
2) *Quality*: react to quality (fitness) factors
3) *Diverse Response*: distribute tasks
4) *Stability*: maintain the group behavior in case of environmental changes
5) *Adaptability*: change the group behavior in case of environmental changes

In summary, each bio-inspired swarm intelligence model stands on its own. There is no common methodology to model swarm algorithms. Therefore, a future step for the CPSwarm project is to define a common approach with a common modeling language for modeling swarm intelligence.

Selected swarm intelligence models, extracted from nature, are presented in the following subsections. They were selected upon two points: i) frequency they are used and thus, their relevance in research, and ii) difference in their methodology.

### 4.1 Ant Colony

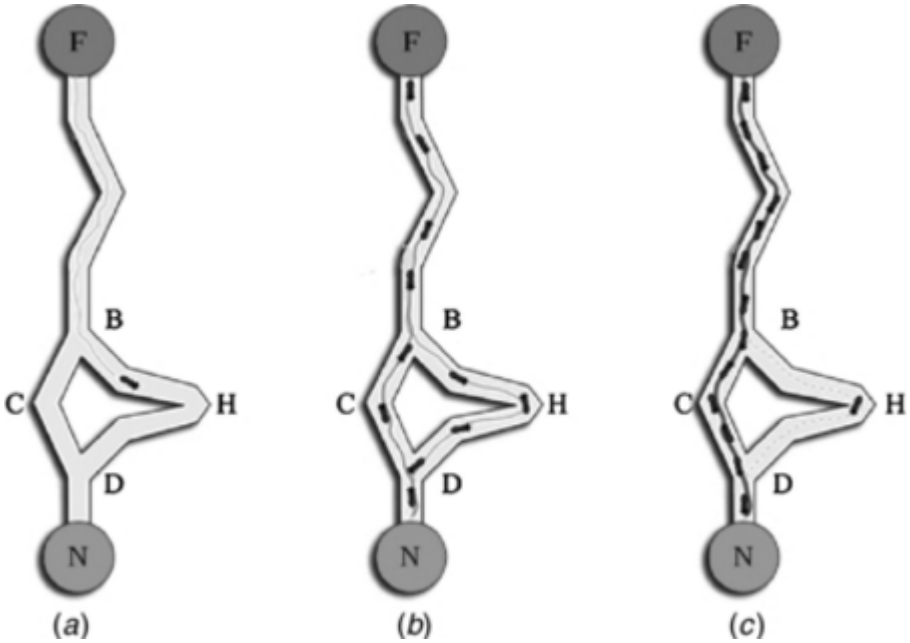



**Fig. 2 Concept of ants finding the shortest route (Chrysostomou, Gasteratos, Nalpantidis, & Sirakoulis, 2012).**

Ants are social insects. They live together in colonies of 2 to 25 million. When the ants perform foraging, they show a typical swarm behaviour. Initially, each ant performs a random search. To find the shortest path to food source from nest, they communicate via laying scent chemicals – so called pheromones – and following them from other ants. Thus, when ants find a food source, they mark the trail from food to the nest. Other ants follow this pheromone trail from nest to food source. Each ant proceeds following the trail, finding food, and again marking the path to the food with pheromones, enhancing the concentration of the pheromones. Finally, the ants follow the pheromone trail with the highest concentration with higher probability, while they are reinforcing the trail with their own pheromone. Thus, the most favourite paths emerge, whereby these are often the shortest of more efficient ones (Yang, Nature-Inspired Metaheuristic Algorithms, 2008).

Main characteristics:

- Distributed intelligence
- Positive Feedback: If an ant follows a pheromone trail, it reinforces the trail – this is called positive feedback: the more ants follow a trail, the higher is the pheromone concentration.
- Stigmergy: A typical characteristic for ants is their indirect communication via the local environment. This process is called stigmergy: a medium is used (the pheromone trails, e.g., on the forest ground). This implies a global data structure.

The main concept of the algorithm can be visualized with following Fig. 2. The ants need to move from nest (N) to food (F) (see (a)). In the given example, two paths are available – a short one (path 1) and a longer one (path 2). First, they start to move on both paths randomly. The probability for choosing path 1 or 2 is equal

with 50% (see (b)). As the ants on path 1 (with point C) come earlier back to their nest, the pheromone concentration is much higher than on path 2. Thus path 1 is subsequently increasing its concentration and thus, its probability. Finally (see (c)), the majority of ants will follow the shorter path 1, as this has a higher probability.

More specific example: Army Ant

The army ants perform their foraging in regular routes with an angle of 123°. If they are not able to find food, they build bivouacs to start with the next 123° on the next day. The amazing behaviour lies in the usage of 123°. Using an even number like 360°/3=120° would lead to a repeated foraging on the fourth day. By using 123°, 10° are left from the first day (see Fig. 3).

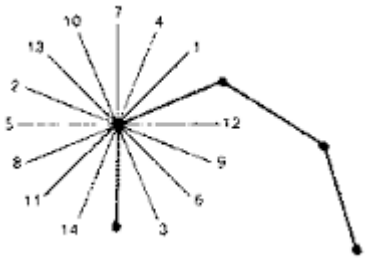



**Fig. 3 Army ants perform a 20-day stationary phase, in which they switch between 14 foraging raids each 123° apart from the last one. The heavy line indicates the colony's path to the new bivouac, where they repeat the 123° separated foraging process.**

### 4.2 Bee Colony

Honeybees are social insects living in colonies of up to 10.000 individuals. They forage food in form of nectar, produce honey and store it in their colony. When honeybees perform foraging, they show a typical swarm behaviour. They use two types of communication: pheromones and "waggle dances". If they are going to be attacked by enemies, they release pheromones to stimulate other bees for attacks. If they are finding a good food source, e.g., a flower patch and bring the nectar back to the colony, they animate other bees with

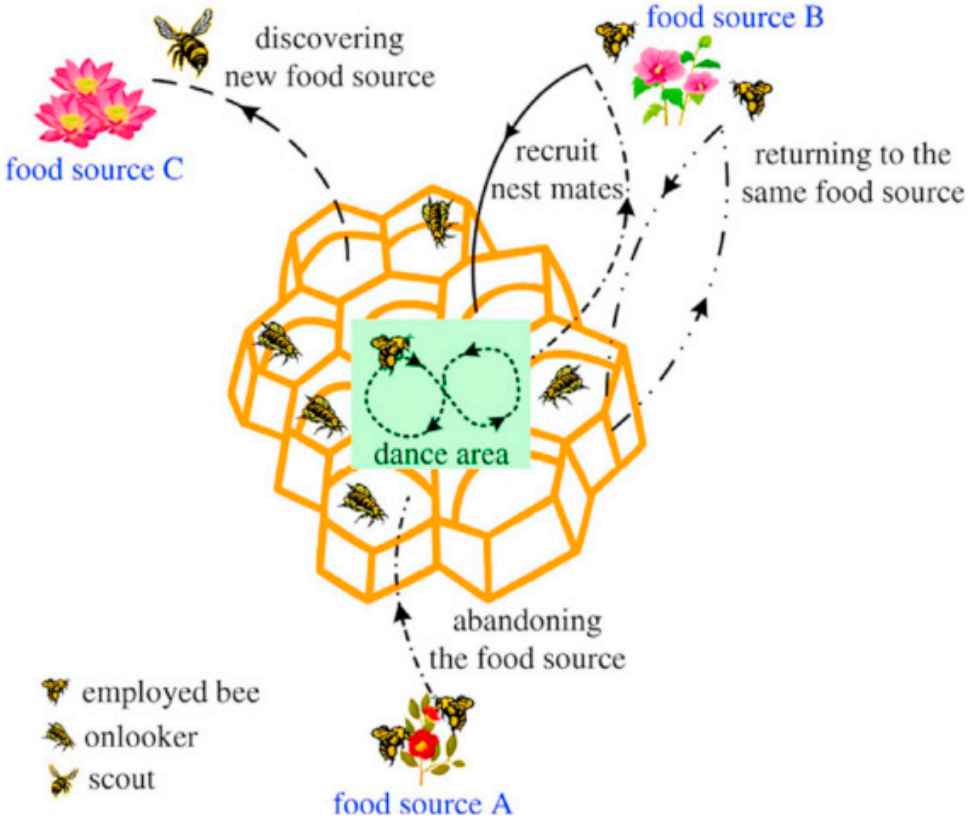



**Fig. 4 Foraging of honeybees (Ghayour, Abdellahi, & Bahmanpour, 2015)**

waggle dances. With the dance performance they communicate the distance with the number of dance runs, and the direction with the angle of runs. The procedure of the waggle dance is illustrated in Fig. 4. In general, we differ between three sorts of bees: employed bees, onlooker, and scouts. Scouts are searching for new food sources in a random way. They return to the hive and evaluate different patches depending on, e.g., sugar content. They deposit the nectar and go to the dance area to perform the "waggle dance" (as employed bees). Onlooker bees make a decision and follow the request. However, the waggle dances differ between types of bees (Yang, Nature-Inspired Metaheuristic Algorithms, 2008).

Another type of collective behaviour can be observed in honeybees' aggregation. They form clusters on different occasions, e.g.,

- In winter to conserve body temperature (see Fig. 5)
- To protect the queen
- When searching new nesting sites
- To stay on the warm side (preferred behaviour of young bees preferring temperature of 36°C)

Main characteristics:

- In foraging: broadcasting ability
- In aggregation: no direct and no indirect communication

**Fig. 5 Example of bee clusters: clustering behavior in winter (Shaun, 2016)**

### 4.3 Fireflies

Fireflies are a family of insects of 2,000 different species. The most of these species produce flashing lights – short and rhythmic. Furthermore, the pattern of these flashes is typically unique to a species. The flashing light (green, yellow, or red with wavelengths from 510 to 670 nanometers) is realized by decomposing a complex carboxylic acid (luciferin) through the associated enzyme luciferase. it is The process of self-light generation is called bioluminescence. Flashing lights have two functions:

- to attract mating partners or
- to attract potential prey (Yang, Nature-Inspired Metaheuristic Algorithms, 2008).

In the attraction of mating partners, females respond to a male's signalling. In some species, females mimic the mating flashing patterns of other species and eat the male fireflies (Yang, 2009).

Another usage of flashing is to inform predators about the bitter taste of fireflies. Tropical species are even able to synchronize their flashes. The rhythmic flash is characterized by the rate of flashing.

**Fig. 6 Swarm of fireflies (©Tsuneaki Hiramatsu)**

In more detail, each male firefly acts as pulse-coupled oscillator (PCO) (see Fig. 7). The oscillator has an internal activation state, Φ(t) that increases constantly over time. Whenever the oscillator state reaches a certain threshold, a pulse is emitted (in the case of Fireflies a brief flash of light) that is observed by the neighbouring agents.

An agent that receives a pulse adjusts its state. In case of an excitatory coupling, the adjustment is always zero or positive. Other models include inhibitory coupling (Klinglmayr, Bettstetter, & Timme, Globally stable synchronization by inhibitory pulse coupling, 2009), where adjustments can be negative or combined excitatory and inhibitory coupling (Klinglmayr, Bettstetter, Timme, & Kirst, 2017). Over time, a connected ensemble of agents synchronizes their phases to each other in order to converge to a synchronous blinking ensemble (Klinglmayr, Bettstetter, Timme, & Kirst, 2017).

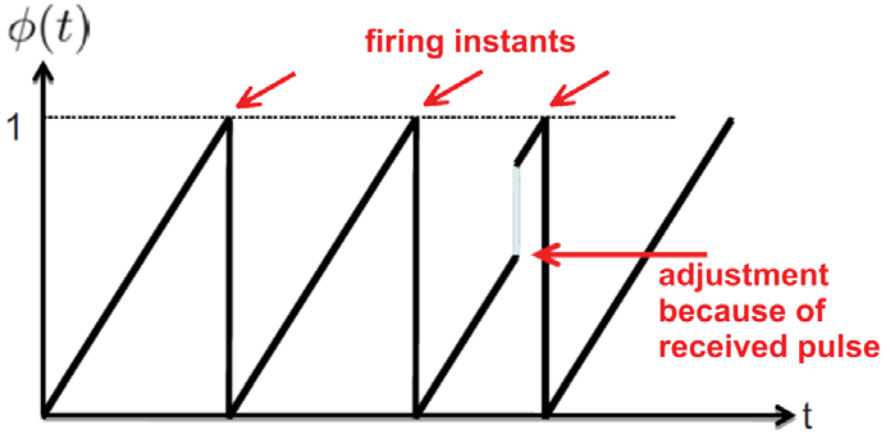



**Fig. 7 Pulse-coupled oscillator (PCO)**

Main characteristics:

- Limited communication distance: The intensity of light decreases as distance r increases. Furthermore, air absorbs light too. This supports the decreased light intensity. The limit distance for communication is several hundred meters at night.

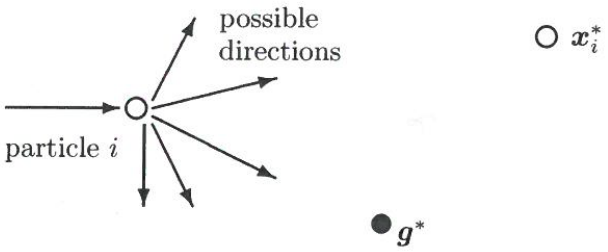
### 4.4 Particle Swarm

The particle swarm optimization algorithm (PSO) was developed by Kennedy and Eberhart in 1995 (Eberhart & Kennedy, 1995). It has no natural model behind, although it has its inspiration in swarm behaviour of fish and birds. Nevertheless, it is much simpler as "real" bio-inspired algorithms or even genetic algorithms as, e.g., it does not use mutation, crossover, or pheromones. PSO is based on two main approaches:

- Real-number randomness
- Global communication among the particles belonging to the swarm

The goal is to find the best location among the local best locations among all particles (or agents)

- within a certain amount of iterations, or
- if the objective is reached.




The main idea is that all particles search the space for an objective by adjusting their trajectories in a quasi-stochastic manner. The movement is a combination between a stochastic and a deterministic component. Furthermore, each particle is attracted by both the global best **g*** and its own best location $\boldsymbol{x}_i^*$ in history. Nevertheless, it still has the tendency to move in a random way. Thus, a particle updates its local best location everytime it finds a better one.

# 5 Swarm Intelligence Algorithms

The following sub-sections present selected swarm intelligence algorithms based upon the selected swarm models of the previous section.

## 5.1 AntHocNet

The AntHocNet algorithm (Di Caro, Ducatelle, & Gambardella, 2004) is designed for routing data packets in a distributed way in mobile ad hoc networks. Although this algorithm doesn't directly use CPSs, it can be used in providing a stable communication among the CPSs. Communication is of high importance in CPS2CPS.
AntHocNet is chosen, as with mobile agents the topology changes constantly and even paths can become inefficient or infeasible. Thus, routing tables need to be updated more frequently. The procedure sets up a path from source s to destination d through a number of nodes i (see Fig. 8, falsely described as graph: in wireless ad hoc networks there are typically no edges. In this figure it simulates the connection.). Each node on the graph selects the next hop for incoming ants (=data packets) based on the local available routing table $\mathcal{T}^i$ (for more information see below).

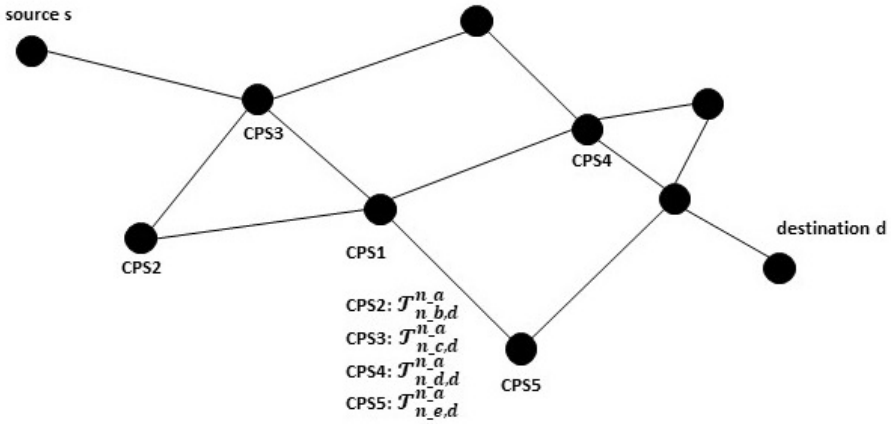



**Fig. 8 AntHocNet: a graph to find a route from source to destination.**

The algorithm uses forward and backward ants (for each type several variants exist, nevertheless, we are going to simplify the main procedure to one forward and one backward ant). As the forward ant reaches the destination, it creates a backward ant, hands over the list of visited notes and terminates itself. The backward ant goes the same way as the forward ant, but in reverse direction so to finally reach the source node. It leaves its pheromones on each node in terms of time it takes to move one hop. This influences the probability at the node to choose a route for the next incoming forward ant.
As the nodes in a mobile ad hoc network move, the relationships between nodes change. Thus, path maintenance and path improvement are important issues.

Advantages of the algorithm:

- Adaptive to network changes
- Robust to ant and node failures
- Provides multi-path routing.

**Issues in using the model of ants in swarms of CPS:**

- The ant as CPS:
  In the AntHocNet, the ant is not the CPS, but a packet sent through the network.

- Initializing the routing table:
  The initialization is done via broadcast: if a destination node d is targeted for the first time, the network is flooded (the packets sent out are called reactive forward ants in the paper (Di Caro, Ducatelle, & Gambardella, 2004)). Reaching destination d, this ant is converted to a backward ant.
- Probability of choosing a route:

$$P_{nd} = \frac{(\mathcal{T}_{nd}^{i})^{\beta_1}}{\sum_{j \in \mathcal{N}_d^i} (\mathcal{T}_{nd}^{i})^{\beta_1}}, \beta_1 \geq 1$$

  where $\mathcal{T}^i$ is the table on node i and, thus, $\mathcal{T}_{nd}^{i}$ one entry representing the "goodness" of the link between the next neighboring node n and the destination d. Furthermore, $\mathcal{N}_d^i$ represents the total number of neighboring nodes the goodness value (or pheromone) $\mathcal{T}_{nd}^{i}$ is stored for. $\beta_1$ is an additional control parameter.

- Dropping pheromones:
  When the backward ant moves through the network it computes the time it takes for one hop $\tau_d^x$ (from the last neighbor to node x). For this computation it includes the accumulation of all those computations to get the value for moving from d to the current node x with $\hat{\mathcal{T}}_d^x$, the number of hops $h$ between x and d, and the time it takes to move one hop $\mathcal{T}_{hop}$.

$$\tau_d^x = \left(\frac{\hat{\mathcal{T}}_d^x + h\mathcal{T}_{hop}}{2}\right)^{-1}$$

  The value $\mathcal{T}_{nd}^{x}$ is then updated for the next forward ant with

$$\mathcal{T}_{nd}^{x} = \gamma \mathcal{T}_{nd}^{x} + (1-\gamma)\tau_d^x, \gamma \in [0,1]$$

- Path maintenance and improvement:
  In regular intervals the nodes broadcast "hello messages". If no "hello messages" are received, e.g., by node a, node b removes the pheromone information from its routing table. If a "hello message" is received, e.g., by node a from node c that was not part of the routing table yet, it is added.
  To improve paths by using "hello messages", a node a adapts the routing tables by inspecting the received contents $\mathcal{T}_{xd}^{x}$ from node x, adds costs for sending a packet from a to x, and creates an estimate $\mathcal{B}_{xd}^{a}$ for sending a packet from a to x.
- Exploration vs. exploitation:
  Already explored routes are assigned with a specific probability. Thus, all ants choose between the available routes – they *exploit* already found routes. Several routes – maybe also shorter or more efficient ones – remain unnoticed. This behavior converges to local optima. Therefore, the ants should choose randomly, with a given probability, etc. to choose between exploitation and further exploration.

**Requirements for CPS:**

- Need a stigmergy – a medium for dropping pheromones, e.g.
  - a digital map (as the presented routing tables)

**Pseudocode**

```
1   //AntHocNet- Pseudocode
2
3
4   //Find routes through forward and backward ants
5   Generate forward ant x_i; //this ant is always generated as soon as the path to d is unknown
6   Select destination node d;
7
8   while (x_i not at d)
9       //Initialize links
10      if (T_nd^i==0)
11          Flood the network with x_i; //this is always performed as soon as the path to d is
12      unknown
13      else
14          for loop over all nodes N
15          Select new neighbor node n with probability P_nd = (T_nd^i)^β1 / Σ_{j∈N_d^i} (T_nd^i)^β1
16          end for
17      end if
18  end while
19
20  Generate backward ant y_i;
21  Terminate forward ant x_i;
22  while (y_i not at s)
23      for loop over all nodes N
24      Drop pheromones τ_d^x = ((T̂_d^x + hT_hop)/2)^-1
25      Update routing table T^i
26      end for
27  end while
28
29  //Link maintenance and improvement
30  //n… neighbours, stored or new, from which a "hello message" was received
31  //m… neighbours, stored in the routing table, where no "hello message" is received
32  if (timer expires)
33      for loop over all nodes N
34          Broadcast "hello message";
35          Receive "hello message"s and store these neigbors n;
36          for loop over all neighbors n
37              if (n in T^i)
38                  Update T_nd^x
39              else if (n not in T^i)
40                  Add n with T_nd^x
41              end if
42              for loop over all neighbors m
43                  if (m not equal to n)
44                      Delete m from T^i
45                  end if
46              end for
47          end for
48      end for
49  end if
```

Commonly known variants to the standard algorithm:

- Elitist Ant System (EAS) – solves route allocation problems

- Max-Min Ant System (MMAS) – solves scheduling problems in software projects
- Rank-based Ant System (RAS) – solves scheduling problems in thermal generator maintenance
- Continuous Orthogonal Ant Colony (COAC) – solves continuous optimization problems
- Recursive Ant Colony (RAC) – estimates parameters of a function

**Applications**

The applications differ, whether the algorithm runs on a central server or distributed on all nodes in the graph (as done in AntHocNet). In the centralized approach, the problem to solve can be observed in god view. In the distributed approach, the problem has several unknowns, e.g., the total number of nodes in the graph or the weights of the links that are known to direct neighbors solely.
Typical problems for ant algorithms comprise:

- Travelling salesman problem
- Protein structure (sequence of amino acids) prediction
- Problems to find the needle in a haystack.

## 5.2 Honeybee Algorithm

The bee algorithm is an optimization algorithm inspired by the foraging behaviour of honeybees, where they use waggle dances to attract other bees.

**Issues in using the model of ants in swarms of CPS** (Yang, Nature-Inspired Metaheuristic Algorithms, 2008)**:**

- Strength of the waggle dance:
  An observer bee follows the dancing bee on a probability, determined with the strength of the waggle dance $w_i(j)$ of bee i at time step t=j
  $$p_i = \frac{w_i^j}{\sum_{i=1}^{n_f} w_i^j}$$
  where $n_f$ is the number of bees in the foraging process. The total number of bees is N, the number of observer bees is N-$n_f$.
- Different roles of bees:
  The natural algorithm describes different types of bee roles: scout, employed bee, onlooker. In typical applications these roles can be mapped onto one agent. Thus, whenever a processing step is finished, the agent just switches its role and performs the tasks related to the new role.
- Fitness value:
  Additionally to direction and angle, the agents need to communicate a fitness value. This fitness value should describe the quality of the "food source" found. Such a fitness value could help other agents to follow the waggle dance request or not.

**Requirements for CPS:**

- Local broadcast technology
- Local memory

**Pseudocode**

```
1
2   //Bee Algorithm - Pseudocode
3   ____________________________________
4   Initialize parameters: scout agents n, sites m, size of patches (incl. site, neighborhood,
5   stopping criterion)
6   while (stopping criterion not met)
7       Select sites for neighbourhood search
8       Recruit agents for selected sites (more agents for best sites)
9       Evaluate fitnesses
10      Select the fittest agent from the patch
11      Assign remaining agents to search randomly and evaluate their fitnesses
12  end                                                                    while
```

Commonly known variants to the standard algorithm:

- Enhanced Bee Algorithm (EBA)
- Grouped Bee Algorithm (GBA)
- Hybrid Modified Bee Algorithm (MBA)

**Applications**

- Function Optimization
- Electronic Design
- Training NN classifiers like MLP, LVQ, RBF and SNNs

### 5.3 BEECLUST

This type of bee algorithm is inspired by the aggregation capabilities of bees: young honey bees form clusters on warm places (Thomas Schmickl, 2011). If they meet other bees, they stop with a certain probability. Their waiting time depends on the local temperature. The colder the local location is, the shorter is the waiting time and the other way around.
After this waiting time, the bee moves on with a random walk.

Advantages for CPS:

- No direct communication among CPSs
- No indirect communication between CPS and infrastructure (stigmergy)
- No memory

**Issues in using the model of ants in swarms of CPS**

- No communication
  This algorithm works without communication (direct or indirect). The agents are somehow selfish and not interested in the other agent's meaning.
- Calculation of the waiting time
  An exemplary calculation of the waiting time is given in (Thomas Schmickl, 2011) with

$$w(r) = \frac{w_{max}\, r^2}{r^2 + 7000}$$

  where their robots measure luminance expressed with r. $w_{max}$ is the maximal waiting time and 7000 is a tuning parameter that needs to be adapted related to the environment the swarm operates in. The tuning parameter can be also modelled as auto-tuning as described in (Kernbach, et al., 2012). w_0 indicates the waiting time if a bee hits a wall.

**Requirements for CPS:**

- Sensor detecting warm places (however "warm places" are specified, e.g., a light/temperature/pressure gradient)

**Pseudocode**

```
1   //BEECLUST – Pseudocode
2
3   Initialize parameters: bees b, waiting times, stopping criterion
4   while (stopping criterion not met criteria)
5       move randomly
6       if b hits a wall
7          stop with waiting time w_0
8       end if
9       if b meets b_j
10         stop with waiting time w(r)
11         w(r) = w_max r^2 / (r^2+7000)
12      end if
13  end while
```

**Applications**

Everywhere, where you have CPS that

- are hard (or even unable) to control
- are able to produce measurements, seldom or inaccurate
- don't know their position (no GPS, no positioning/localization system available)
- don't have memory
- don't have a communication ability

Examples are passive agents like intelligent buoys, fish trap, tank and pipe systems.

### 5.4 Firefly Algorithm

The firefly algorithm is a multimodal optimization algorithm inspired by the light flashing of fireflies (Yang, 2009). The main idea is to attract other fireflies for aggregation.

**Issues in using the model of fireflies in swarms of CPS**

- Same type (gender):
  It is assumed that each CPS is of the same type. Biologically speaking, there are all unisex. Thus, each firefly is attracted by others independent of their type.
- Attractiveness:
  The value for attractiveness β is proportional to their brightness I. For example, if two fireflies i and j flash, the less bright flashing one will move to the brighter flashing one. Furthermore, both values decrease if distance r between them increases. In the simplest form, the light intensity is expressed bv inverse square law $I(r) \approx \frac{1}{r^2}$. To avoid singularity with r=0, and a fixed light absorption coefficient γ, it can be approximated with the Gaussian form
  $$I(r) = I_0 e^{-\gamma r^2}$$
  where $I_0$ is the light intensity at r=0. The absorption coefficient γ typically varies from 0.1 to 10 starting with the characteristic length
  $$\gamma = \frac{1}{\Gamma_m}, \Gamma_m = \gamma^{-1/m} \to 1, m \to \infty.$$
  As the attractiveness β is proportional to the light intensity, we can define it with
  $$\beta(r) = \beta_0 e^{-\gamma r^2}.$$
  Furthermore, we define the distance $r_{ij}$ between firefly i and firefly j with the Euclidean distance
  $$r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

A random walk biased towards the brighter fireflies could have following form

$$\boldsymbol{x_i} = \boldsymbol{x_i} + \boldsymbol{\beta_0 e^{-\gamma r^2}}\left(\boldsymbol{x_j} - \boldsymbol{x_i}\right) + \boldsymbol{\alpha \epsilon_i} \quad 1$$

where $\boldsymbol{\epsilon_i}$ is a random vector uniformly distributed in [0,1] and $\alpha \in [0,1]$ is the randomization parameter.

**Requirements for CPS:**

- Sensing capabilities for light

**Pseudocode**

```
//Firefly Algorithm – Pseudocode

Generate initial population of fireflies x_i (i=1,2,…,n)
Light intensity i_i at x_i is determined by f(x_i)
Define light absorption coefficient γ
while (t<MaxGeneration)
for i=1:n all n fireflies
    for j=1:n all n fireflies
        if(I_i<I_j)
           Move firefly I towards j by applying equation 1;
        end if
        Vary attractiveness with distance r via exp[-γr]
        Evaluate new solutions and update light intensity
    end for
end for
Rank the fireflies and find the current global best g*
end while
```

Commonly known variants to the standard algorithm:

- Discrete firefly algorithm (Durkota, 2011)
- Binary firefly algorithm (Falcon, Almeida, & Nayak, 2011)
- Chaotic firefly algorithm (Zhang & Wu, 2012)
- Parallel firefly algorithm (Subotic, Tuba, & Stanarevic, 2012)
- Lèvy flight firefly algorithm (Yang, 2010)
- Gaussian firefly algorithm (Farahani, Abshouri, Nasiri, & Meybodi, 2011)

**Applications**

Typical fields for firefly algorithms comprise making and supporting decisions in the field of engineering, computer science and communication including

- Cluster head selection
- UCAV path planning
- Job scheduling

### 5.5 Particle Swarm Optimization (PSO)

In the particle swarm optimization (PSO) we would like to find the best solution with n particles, whereby all of them perform quasi-random walks in a stochastic manner by positional vectors.

## Issues in using the model of PSO swarms of CPS

- Velocity of particles:
  The velocity of particles $\boldsymbol{v_i}$ for particle i is determined with

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \alpha\boldsymbol{\epsilon}_1 \odot [\boldsymbol{g}^* - \boldsymbol{x}_i^t] + \beta\boldsymbol{\epsilon}_2 \odot [\boldsymbol{x}_i^* - \boldsymbol{x}_i^t]$$

where $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are random vectors, whereby each entry is a value in [0,1]. Furthermore, α and β are the acceleration constants and both typically have a value of 2. A typical starting value for the velocity at t=0 is $\boldsymbol{v}_i^0 = 0$. The velocity could be anything, nevertheless, the value is typically bounded by a range $[0, \boldsymbol{v}_{max}]$.

**Pseudocode**

```
//Particle Swarm Optimization Algorithm – Pseudocode

Objective function f(x), x=(x_1,…,x_p)^T
Initialize locations x_i and velocity v_i of n particles
Find g* from min{f(x_1,…,f(x_n)} (at t=0)
while (criterion)
    t=t+1 (pseudo time or iteration counter)
    for i=1:n all n particles and all d dimensions
        generate new velocity v_i^t+1
        calculate new locations x_i^t+1=x_i^t+v_i^t+1
        evaluate objective functions at new locations x_i^t+1
        find the current best for each particle x_i*
    end for
    find the current global best g*
end while
```

Commonly known variants to the standard algorithm:

On overview of all PSO variants can be found in (Imran, Hashim, & Khalid, 2013). In general, researchers adjust the algorithm in terms of introducing constriction factors, inertia weights or differ in the initialization of particles.

**Applications**

- Applications for PSO have a wide range, mainly including multimodal problems, or problems for which no specialized methods are available (Poli, 2008). These include, e.g., Antenna design
- Design and optimization of communication networks
- Clustering, classification, and data mining
- Combinatorial optimization problems
- Design of controller

# 6 Initial Approach: Common Modelling for Swarm Intelligence Algorithms

As we have seen in Section 4: There is no common methodology to model swarm algorithms. The modelling of swarm algorithms extracts behavior of the nature and abstracts it to the technical domain (for the entire procedure see Fig. 1). Nevertheless, there is no standard modelling methodology, technique or language to visualize these models. Moreover, it is hard to combine different swarm behaviors from different sources.
This challenge is picked up by the project CPSwarm in WP4: We introduce a common modelling standard for swarm behavior. The main idea is to have a formulation and definition of models visually representing in SysML i) swarm intelligence algorithms (Section 6.2) and ii) individual behavior of existing swarm intelligence algorithms to model new ones (Section o). In both cases, the final goal is to deposit code to those models and model pieces to speed up the development and reusability process.
The developed swarm models are stored on the Modelio Forge available at the following web link https://forge.modelio.org/projects/cpswarm-modelio36/files.
Installation instructions for the modules to run the example are included in D5.2.

First, the process of modelling swarm intelligence algorithms is formalized in Section 6.1.

## 6.1 Formalization of models for swarm intelligence algorithms

The formalization of the models is adopted from "D4.1 – Initial Catalogue of CPS" models. Therein, each model (swarm member, environment, and goal) has following three components:

1) Unique name:
   - each model needs to be distinguishable from other models by name
   - each model's name needs to be given in a way that it is associated with the model's functionality

2) Description:
   - a detailed description is necessary for i) documentation and ii) the programming tasks by the software developer
   - the description is created as parameter of the model with string [256]

3) Parameters:
   - each model contains a set of parameters that have the following form:
     - Property: name, *type* [*range*]: Defines a constant parameter of the model
     - Input: name, *type* [*range*]: Defines an input parameter to the model
     - Output: name, *type* [*range*]: Defines an output parameter of the model

## 6.2 Library of swarm intelligence algorithms

This library comprises the given swarm intelligence algorithms as presented in Section 5 from a high-level view. In the final modelling library, they can be found ready to use, including
- a description of their functionality
- defined inputs and outputs
- defined local states (if necessary)
- deposited Java and C++ code (in a first version: pseudocode)

On the example of the swarm algorithm BEECLUST (see Section 5.3 for more details) we model a swarm member. This includes the swarm algorithm as well as corresponding sensors and actuators. The modelling is visualized in Fig. 9, and following models are used:

Models of the library:

- "bc: BEECLUST"
  - Description: "The BEECLUST is used for clustering of swarm individuals applying 3 rules: 1. Move randomly,2. If a bee meets another bee, they stop with a certain probability. Their waiting time depends on the "local temperature". The colder the local location is, the shorter is the waiting time and the other way around. 3. After this waiting time, the bee moves on with a random walk."
  - LocalState: w_0, type: int
  - LocalState: wmax, type int
  - Input1: poi:Point2D, type: int [2]
  - Input2: luminance, type: float
  - Output: pos:Point2D, type: int[2]
  - Pseudocode:

```
move randomly
if swarm_member hits a wall
    stop with waiting time w_0
end if
if swarm_member meets other swarm_member
    stop with waiting time w(r)
    w(r) = (w_max r^2) / (r^2+7000)
end if
```

- "ps: POISensor"
  - Description:"The POI sensor returns a vector, describing the path to the next point of interest (POI)"
  - Output: poi:Point2D, type: int [2]
- "ls: light sensor"
  - Description: "The light sensor measures the luminance"
  - Output: luminance, type: float
- "l: Locomotion"
  - Description: "The locomotion motor moves the agent to a given x/y coordinate."
  - Input: pos:Point2D, type: int [2]

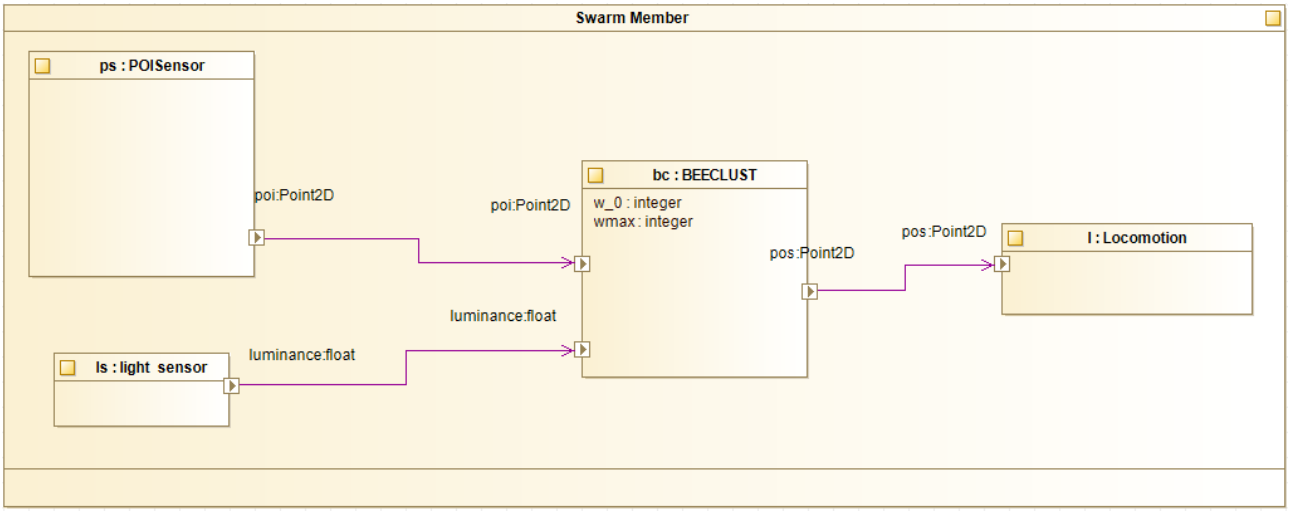



**Fig. 9 Model to represent a swarm member with BEECLUST.**

### 6.3 Customizable swarm intelligence algorithms

Typically, real-world applications come with needs that cannot be directly modelled with an existing nature-inspired swarm intelligence algorithm. Therefore, it is useful to have a process that allows constructing customized swarm intelligence algorithms. This is enabled by a library that provides single behaviors extracted out of given swarm intelligence algorithms.

In the modelling tool, you can construct a state machine with all the behavioral elements you need (a selection of those elements in the library is visualized in Fig. 11). In the final version, java code will be deposited to each state.

Furthermore, each behavioral element (= state in the state machine) has a number of inputs and outputs that allow to connect with other elements or with components of the modelled swarm member (e.g. sensors – see D4.1). This state machine can be summarized to a "global" model MySwarm (see Fig. 10), which can be used as behavior for the swarm member (as done with the BEECLUST in Fig. 9). Therein, you can define the unique name, the description, the number and type of inputs and outputs, and local states. An exemplary customized swarm algorithm is depicted in Fig. 12. This customized swarm algorithm is inspired by the BEECLUST. We additionally add the state "Update routing table" to the BEECLUST functionality.

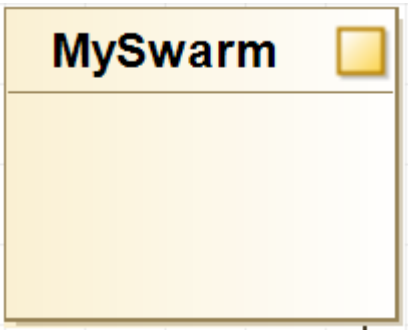



**Fig. 10 Model of a customized swarm intelligen algorithm (high-level view).**

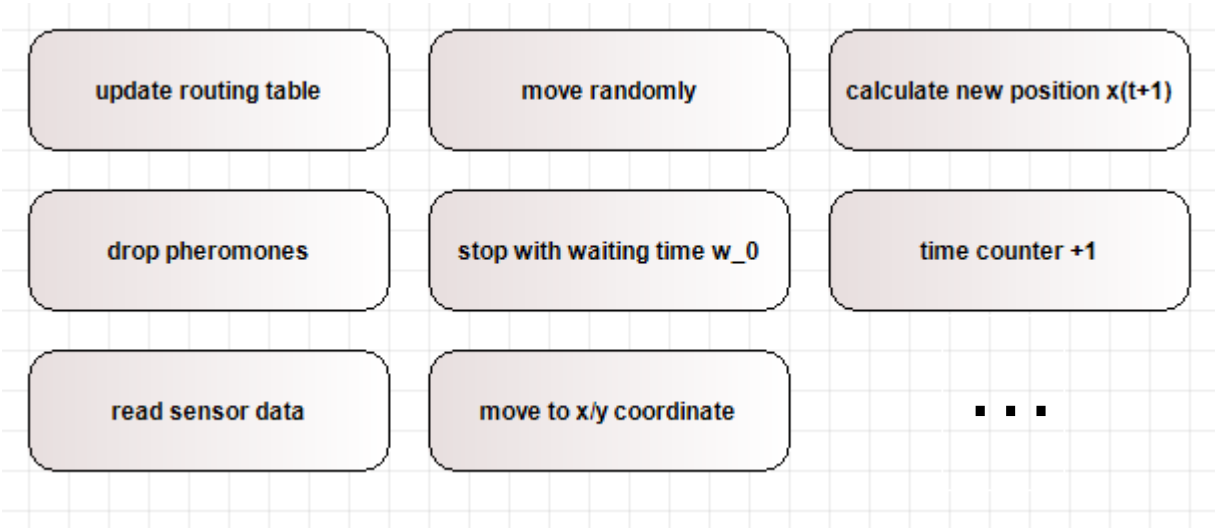



**Fig. 11 Selection of available states in the library.**

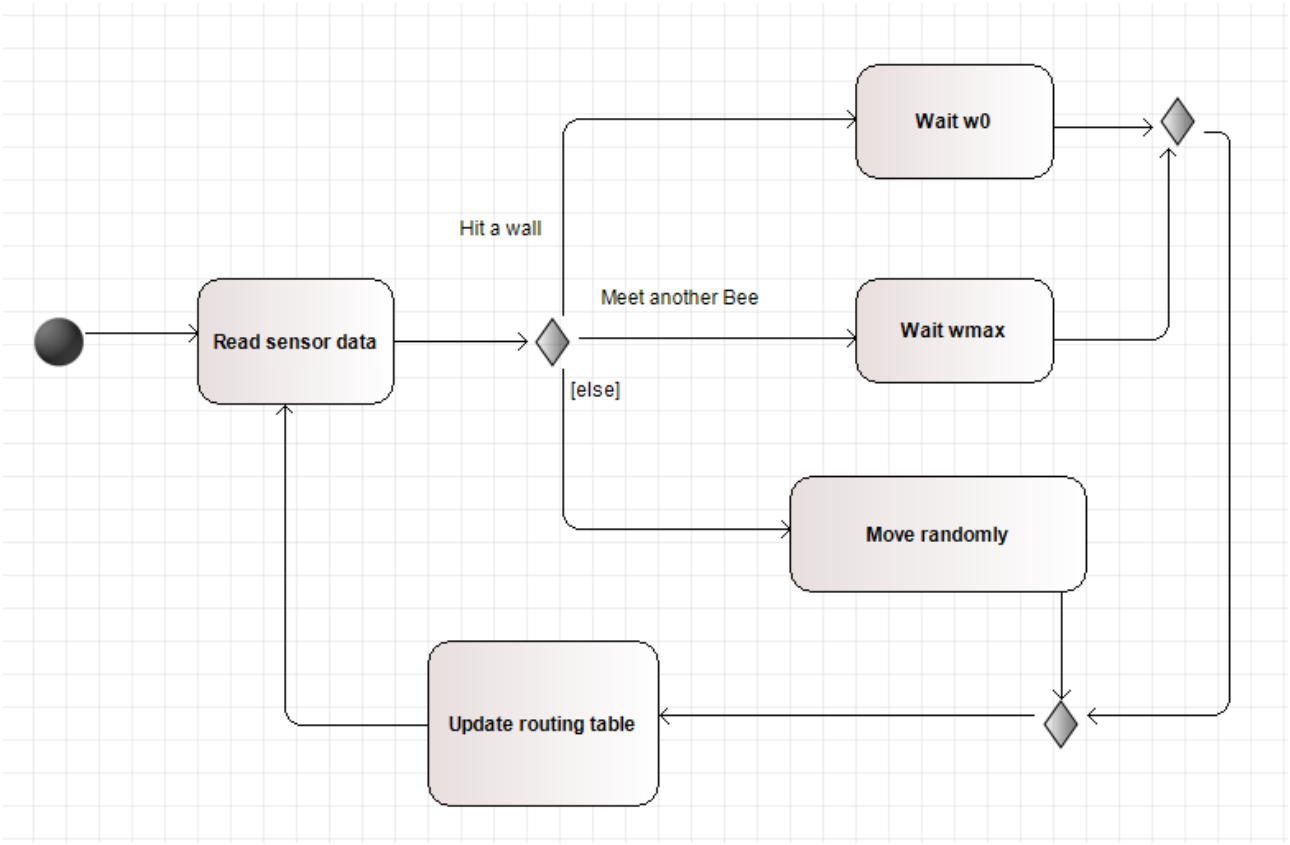



**Fig. 12 Model of a customized swarm intelligen algorithm (low-level view).**

# 7 Conclusions

In this deliverable D4.4 we provided an overview on swarm intelligence algorithm from the state-of-the-art. First, we described the natural models behind them. Then we took a closer look on the algorithmic side, by focusing on CPS relevant topics, including requirements and issues for swarms of CPSs, and where they can be applied. Finally, we presented the current state of developing a common swarm modelling approach on two levels: high-level – where a model describes an entire swarm algorithm, and low-level – where the modeler is able to customize its own swarm algorithm through mini-behaviors.

In future work, the swarm intelligence algorithms will be segmented to application specific properties. This will enable a directed application of swarm intelligence algorithms to the use cases envisioned in the CPSwarm project.
Additionally, following state-of-the-art models will be processed, and documented in the next deliverable D4.5:

- Synchronization of swarms of fireflies
- Bird flocks
- Bacteria swarms
- Fish schools
- Quadruped herds
- Evolutionary Models

Moreover, Human2Swarm recipes will be introduced. A comprehensive introduction to this topic can be already found in D4.1

## Acronyms

| Acronym | Explanation |
|---|---|
| DoA | Description of Action |
| KPI | Key Performance Indicators |
| CPS | Cyber-Physical Systems |
| PSO | Particle Swarm Optimization |

## List of figures



## List of tables

No tables.

## 8 References

Ahmed, H. R., & Glasgow, J. I. (2012). *Swarm Intelligence: Concepts, Models and Applications.* Kinston, Canada, Queen's University, School of Computing Technical Reports.

Chee Peng, L., & Dehuri, S. (2009). *Innovations in Swarm Intelligence.* Springer-Verlag Berlin Heidelberg.

Chrysostomou, D., Gasteratos, A., Nalpantidis, L., & Sirakoulis, G. C. (2012). Multi-view 3D scene reconstruction using ant colony optimization techniques. *11*(23).

Di Caro, G., Ducatelle, F., & Gambardella, L. M. (2004, September). AntHocNet: An Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks. (L. N. Science, Ed.) *Parallel Problem Solving from Nature*(3242), pp. 461-470.

Durkota, K. (2011). Implementation of a discrete firefly algorithm for the QAP problem within the SEAGE framework. *Czeck Technical University in Prague, Faculty of Electrical Engineering*.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39-43.

Falcon, R., Almeida, M., & Nayak, A. (2011). Fault Identification with Binary Adaptive Fireflies. *IEEE Congress of Evolutionary Computation*, 1359-1366.

Farahani, S., Abshouri, A., Nasiri, B., & Meybodi, M. (2011). A Gaussian Firefly Algorithm. *International Journal of Machine Learning and Computing, 516*, 448-453.

Ghayour, H., Abdellahi, M., & Bahmanpour, M. (2015). Artificial intelligence and ceramic tools: Experimental study, modeling and optimizing. *Ceramics International*(40), pp. 13470-13479.

Green, D., Aleti, A., & Garcia, J. (2017). The Nature of Nature: Why Nature-Inspired Algorithms Work. In S. Patnaik, X.-S. Yang, & K. Nakamatsu, *Nature-Inspired Computing and Optimization* (pp. 1-27). Springer.

Hamann, H., & Schmickl, T. (2012). Modelling the swarm: Analysing biological and engineered swarm systems. *Mathematical and Computer Modelling of Dynamical Systems, 1*(18), 1-12.

Imran, M., Hashim, R., & Khalid, N. E. (2013). An Overview of Particle Swarm Optimization Variants. *Procedia Engineering*(13), 491-496.

Kernbach, S., Häbe, D., Kernbach, O., Thenius, R., Radspieler, G., Kimura, T., & Schmickl, T. (2012). Adaptive collective decision-making in limited robot swarms without communication. *The International Journal of Robotics Research, 1*(32), pp. 35-55.

Klinglmayr, J., Bettstetter, C., & Timme, M. (2009). Globally stable synchronization by inhibitory pulse coupling. *2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies*.

Klinglmayr, J., Bettstetter, C., Timme, M., & Kirst, C. (2017, April). Convergence of Self-Organizing Pulse-Coupled Oscillator Synchronization in Dynamic Networks. *IEEE Transactions On Automatic Control, 62*.

M. Dorigo, V. M. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26*(1), pp. 29-41.

Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, pp. 1-10.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, (pp. 25-34 ).

Shaun. (2016, November 27). *What happens to honeybees in winter*. Retrieved from Cox's Honey - Let us bee your honey!: https://coxshoney.com/what-happens-to-honeybees-in-the-winter

Subotic, M., Tuba, M., & Stanarevic, N. (2012). Parallelization of the Firefly Algorithm for Unconstrained Optimization Problems. *Latest Advances in Information Science and Applications*, 264-269.

Thomas Schmickl, H. H. (2011). BEECLUST: A Swarm Algorithm Derived from Honeybees. In Y. Xiao, *Bio-inspired Computing and Networking* (pp. 95-137). CRC Press.

Yang, X.-S. (2008). *Nature-Inspired Metaheuristic Algorithms.* Luniver Press.

Yang, X.-S. (2009). Firefly Algorithms for Multimodal Optimization. (I. S. Algorithms, Ed.) *Stochastic Algorithms: Foundations and Applications*, pp. 169-178.

Yang, X.-S. (2010). Firefly Algorithm, Lévy Flights and Global Optimization. *Research and Development in Intelligent Systems XXVI*, 209-218.

Zhang, Y., & Wu, L. (2012). Rigid Image Registration based on Normalized Cross Correlation and Chaotic Firefly Algorithm. *International Journal of Digital Content Technology and Its Applications, 6*(22), 129-138.