Prevention of Ant Mills in Pheromone based Swarm Algorithms

Ahmad Reza Cheraghi, Jochen Peters, Kalman Graffi

Technology of Social Networks, Heinrich Heine University Düsseldorf, Germany
Honda Research Institute Europe GmbH, Offenbach am Main, Germany
ahmad.cheraghi@hhu.de, jochen.peters@hhu.de, kalman.graffi@honda-ri.de
https://tsn.hhu.de, https://www.honda-ri.de

***Abstract*—Technical developments are often motivated by concepts from nature. The ant algorithm is such a concept. This algorithm takes on the idea of using the environment as a data store in order to contact to other participants (robots or ants). Copying this concept, however, the same problems occur, like real ants have. Especially with the army ant, which is characterized by their uniform behavior, ants mills appear. Gadget by a wide pheromone trail an ant on a random circle intensifies this trail for each round. Other ants that run in these the circular trail, they additionally reinforce it until an ant mill is formed and the whole colony starves to death. Nature solves this problem with a new breed at a new location. In an implementation as mars robots or molecular machines a new breed is not possible. We develop a closer look at this ant mill problem. We also find the ant algorithm interesting in terms of molecular machines where a wireless link is not an option for communication. In our simulation the ants build randomly ant mills and highly frequented routes to already depleted sources of food. Both cases are characterized by a high density of ants. Therefore, we present an extended algorithm which triggers an escape behavior, if three or more ants are standing near to each other. Evaluation shows a significantly better food supply than without this modification leading to a reduction in ant mills.**

***Index Terms*—multi-robot systems, simulation, pheromones, insects, molecular machines**

I. Introduction

Ants use pheromones (scents) as information carriers to organize their colony (see Blum [1] and Fletcher [2]). Ants, for example, use special pheromones to mark distances, which was investigated in more detail in Tentschert [3]. The ant algorithm takes up this idea from nature. For example, if an ant just comes from a food source with high nutritional value, it tries to communicate this by a correspondingly strong pheromone label in its environment. If another ant comes upon this mark, it makes a decision based on its own running direction, sensors and other perceived markings in its environment. This decision is also recorded as pheromone release in the surrounding area. While without such a marking the running behavior of the ant is random, structures form in the environment with marking. Desirable structures are e.g. ants, which oscillate between food and nest, and thereby strengthen this trail by their pheromones. It is also desired that a part of the ants search for new sources of food by a random running behavior. An undesirable structure, however, is the ant mill, in which an entire colony can die. All ants run in a spiral or circle instead of exploring the surroundings or collecting food. This problem is a well known in nature. In 1944 the American Museum of Natural History (New York City) described them very detailed in [4] and Delsuc [5] (2003) ask the question, why ant mills (part of the "army ant syndrome") are not eliminated by the evolutionary history.

The ant algorithm is coming without a radio connection and is independent of the number of ants. Besides, an ant only remembers a little information: some data to find the nest (which could also be marked with another kind of pheromone) and which food source (biggest nutritional value) the ant has already experienced in order to create a pheromone output based on a comparison between the food sources. Two further information do not require any storage in a hardware solution: The current direction of movement and whether the ant carries straight food or not. Because of its simplicity, we think the algorithm – or rather the concept – can be applied to molecular machines in a modified form. The idea behind molecular machines is a big molecule acting like a complex biological protein. We found small molecular machines acting like a logical AND [6], adder [7], robotic arms [8] and other machine parts in Balzani [9] and Jones [10]. It is only a question of time, that these molecular machines are able to work in a robot swarm. Keeping this idea in mind, we need a simple logic for each "machine" without traditional wireless network communication. In our simulation we followed the simple approach of the ant algorithm, and it turned out that ant mills are hard to ignore. As a solution we offer that an ant shows an escape behavior near at least three other ants. We were able to demonstrate a better food supply to the colony and a reduction of ant mills.

In the following section we describe how other works formulate and apply the ant algorithm. Among them are also works that deal with the tracking accuracy of ants or the probability of finding food. Our combination of these topics, including a case in which no pheromones initially lead to food, has not yet been found in the literature. In section III we describe our simulation and all important components. This includes all parameters that influence the simulator and the algorithm. We describe the problem of the ant mill and how our modification solved this problem. We also describe our implementation of the ant algorithm and the small escape

mechanism to avoid ant mills. The measured values are described, too, e.g. the development of the average age of an ant in the colony or the distance covered by an ant before its death. Section IV describes the results of our simulation. It is explained how this, together with the amount of food collected, can describe the success of our modification. The last chapter summarizes our results. We also consider new questions regarding the simplifications of the ant algorithm as well as areas that we have not yet considered with our simulation.

II. Related Work

In the following section we will look at existing work on the ant algorithm and what the algorithm was used for. In addition, we have selected works that deal with the finding of food and the formation of ant mills. In our research we looked for a uniform description and use, but could not find a "classical" ant algorithm. Additionally we couldn't find related work that deal with all aspects for a practical application.

The first ant algorithm is documented at "Distributed optimization by ant colonies" [11] in 1991 and they tried to find a heuristic to solve the traveling salesperson problem (TSP). Another work was focused on lane formation by Couzin [12] in 2003. Both sets simulated ants on existing trails and analyzed the movement. Couzin tried to understand in which situation ant mills are build in respect to the strength of following a trail. Couzin compared the simulated ant movements with movies from real army ants. The probability of finding a single food location is handled by Li [13] in 2014. Li adds aspects like the ant speed depending on the age of an ant: Simulated ants with a high age have a bigger radius around the nest, to find it. Related to Sommer [14], desert ants are able to store a small number of routes next to a location. The sandy surface removes pheromone marks very fast away and the desert ants using eyes and brain primarily to re-find a location.

These works had in common that ants released pheromones, reacted to it and pheromones lost their intensity over time in the environment. Except for Li [13] the pheromone trails are already initially defined, but Li does not treat ant mills. Couzin's work deals with it, but not the aspect of efficient foraging. In the first publication from 1991 the pheromone trails are fixed as a graph, and Li does not. While Li does not mention the algorithm for the ant's decision, in the other works this is a result of linear optimization. There are also differences as to whether ants constantly release pheromones or only when they run from a food source into the nest. Nobody used a uniform scenario or a 100% comparable ants algorithm. Furthermore, from our point of view, not all aspects for successful practical application are simulated, such as the problems caused by ant mills or the lack of initial pheromone trails.

We wanted to implement an ant algorithm that collects food and gets as many ants as possible to the food sources found by other ants by random walk. From our point of view, this is the main task of the ant algorithm. After implementing a "classic" ant algorithm (see section III), we were confronted with ant mills, which resulted in a high mortality rate in the colony and a poorer food supply. We found no work that simulates a complete ant colony and treats or tries to solve these aspects. We have therefore come up with a solution to the ant mill problem to make a practical application of the ant algorithm possible.

In our work we added a kind of traffic control to the ant algorithm, so that the ants can still orient themselves on pheromone trails, but areas with "mass-meetings" cannot arise. This small modification has greatly improved the collection of food.

III. Methods

In this section we first roughly describe what we understand by a "classic" ant algorithm and how we have modified/extended it. This is followed by a description of the parts of our simulation and how it works. The algorithm that each ant uses when running the simulation is explained by small examples. Initial constants are described as well as parameters that we varied when we started a simulation.

In section II we tried to make clear that the ant algorithm has to be considered much more than a concept, as a clearly defined general algorithm. This is mainly due to the depth of implementation, how detailed one wants to depict the nature of the ant and an ant colony. This can lead to very different solutions for different questions. It cannot be ruled out that the work we found used only the scenarios for the ants, in which no "misbehavior" like ant mills occurs. We understand the ant algorithm as a method that modifies the random running behavior of ants in such a way that frequently visited positions are increasingly visited by ants. In detail: If an ant encounters a food source, it runs with the food directly into the nest and places a pheromone trail (homing mode). This trail depends on the nutritional value of the food source. If an ant that is not in "homing mode" encounters pheromones, the ant does a score. Two aspects are considered in this score: the strength of the pheromone found and the running direction of the ant. By this scoring the ant decides for a new running direction and increases the pheromone at the position, for whose direction it decided. Ants in "homing mode" also change the pheromones, but only on the fields they cross. They also score the pheromone trail based on their experience of how nutritious the food source they come from is. As soon as an ant has arrived in the nest, it leaves the "homing mode". A memory where the food source is, therefore, exists only as a laid pheromone trail. In this ant algorithm, which we call "classic", the strength of the placed pheromones also decreases over time. We implemented this classic form of algorithm and got many small ant mills as a result.

A. Prevention of ant mills

Ideally, the strongest pheromone trails should be the way between food and nest, because an ant places this trail exactly like that. If a second and third lane is laid parallel due to the size of the food area, the path between nest and food is no longer clear. Similar to large distributed systems, the "misbehavior" of an ants running a circle in a wide pheromone lane is not the exception, but the rule! For the prevention of ant mills, we once again remembered the basic idea behind the route formation. A single ant did not try to recognize a route because it had neither enough data nor enough processing logic. We thought that we could not recognize countless ants running in circles with a single ant. We have limited ourselves to saying that an ant should prefer positions with strong pheromones, but to prevent a mutual reinforcement of a group of ants, our modified ants must avoid ant groups. We have therefore triggered a escape behavior of the ant from a number of 3 neighboring ants. This behavior is a running forward in a randomly selected direction of the ant and stops after a short time. We do not care whether the ant escapes from a trail used by many ants to reach food, or a nasty ant mill. The ant cannot recognize either anyway and our simulations show that this forced scattering of ants makes the classical part of our ant algorithm much more successful.

B. Implementation

During a concrete implementation we paid attention to being able to easily modify these ant algorithm. For comparison we have programmed three variations:

classic This is the method described above and explained in more detail in the last subsection.

modified It is like the classic one, but if there are more than 3 ants nearby, the ant escapes for 5 rounds in a random direction.

random No pheromones or escape behavior are used.

To analyze these variants we have done a simulation with the following components:

- a simulator (running 1000 rounds)
- a world (2 dimensional)
- food with a nutritive value
- pheromones with a potency
- 30 ant objects (each with an algorithm, we want to test and compare)

We now describe this in more detail.

1) Simulator: The Simulator loads a world (generated by a seed) and loop 1000 rounds through 30 ants. Thus each ant object runs our algorithm for 1000 times. We call this a wake-up call because the simulator awakens an ant to act only once in each round.

2) World: Our world is a 2 dimensional hex grid. Thus our ants, food and pheromones can only placed into that grid. The worlds are generated with 10 different random seeds to place food with different area sizes, positions and nutritive values. When we measured a new result with a different parameter, we simulated it with the same 10 different worlds and build a cumulative value.

This hex grid is placed on a normal (x, y) coordinate system, but the placement of everything must fit into the hex grid. If we talk about distances we use the euclidean distance based on that x and y values. Because of the hex grid, the ant can only move to 6 neighbor fields. For Example if the ant starts at $(0, 0)$ and is awoken by the simulator, it can make a step to $(-1.0, 0.0)$, $(1.0, 0.0)$, $(-0.5, 1.0)$, $(-0.5, -1.0)$, $(0.5, 1.0)$ or $(0.5, -1.0)$.

3) Food: Food is part of a food area. In a world generated by a seed these areas can have a size between 1 and 16 grid locations. A world can have up to 30 areas. Each food is generated with a nutritive value between 1 and 20. If an ant reaches the food, the nutritive value is decremented by one. If the nutritive value is zero, the food is deleted immediately. The initial placement of the areas is handled by a random seed. There are two different concepts placing them with a maximal area distance of 20 ($\pm$ 5): spray and ordered. The spray concept sets an area on a random position and places the next area randomly (but not to far away). Finally the sprayed areas are centered in the middle of the world. The ordered concept is a grid arranged around the ant nest. The position is varied by 5. We decided to choose these concepts to get random worlds with reachable food for the ants.

4) Pheromones: In our classical and modified variation of the ant algorithm pheromones are used. The pheromones are generated by ants, if an ant finds food and runs back to the nest. The ants are able to increment the pheromone potency. A pheromone with a *potency* less than 0.0001 is removed by the simulator. The maximum value of a pheromone potency is 1.0. If this value is reached, nothing more can be added.

After all ants do their work in a simulator round, the pheromones are deleted or the potency is reduced by an *evaporation*:

$$potency_{new} = \frac{100 - evaporation}{100} * potency_{old} \quad (1)$$

The evaporation was defined by us before each simulation. We have tried different evaporation values. We thought, for example, that a high evaporation gives the weak trail of a single ant less weight. So if a single ant creates a circle at random, it loses its pheromone potency more quickly than a pheromone trail that is continuously created by several ants from a food source to the nest.

5) Ant: We initialized the simulator with 30 ants placed on $(0, 0)$ in the world (we call it *nest*). Thus they start with a traffic jam. Each ant has a *hunger counter*. If an ant does not reach any food or carries it to the nest, the hunger counter is increment by one. If the limit reaches 50, it dies immediately and respawns at the nest. Every ant is awoken by each simulator round and runs our algorithm. That's why there were always exactly 30 ants in the world. In addition to the variables necessary for the algorithm, we stored the number of rounds we have experienced for each ant: this is the *age*

of an ant. We used an ant's starvation to set that age back to zero. A high average age of ants is therefore an indicator of how well the ant colony is supplied with food.

Besides the *age* of the ant we also stored the *distance* which was between the position where it dies and the position 10 simulation rounds before. In our simulation we only came across ant mills with a diameter of less than 4 grid cells. Now if many ants die without having traveled a greater distance before, we assume an ant mill. In the section IV we go into this indicator once more precisely.

C. Our tested ant algorithms

We have developed an ant algorithm, which from our point of view best corresponds to a classical ant algorithm. We have already explained the problem of a unique Ant algorithm in section II, since the application and degree of mapping of a natural ants behavior can be different. We have added a *modification* to our algorithm with which we can trigger an ant to escape. For the sake of simplicity, we have focused in this work on triggering the escape behavior of 3 or more ants in the vicinity of an ant. To switch off this modification, we have set the value to 31. Since only 30 ants exist in our simulation, no escape behavior takes place at a value of 31; this is what we call the *classic* ant algorithm. For a better comparison we have tested a *random* ant algorithm. This corresponds to our classic one, however, the use of pheromones is completely switched off. In the random ant algorithm there is therefore no interaction or detection between the ants: neither by pheromones nor by escape behavior.

We want to make it clear at this point in advance: Our simulation and each ant algorithm tested by us does not generate a certain number of ant mills by direct influence. Our classic ant algorithm is also not able to detect a trail to the food but reacts only to a few pheromones strengths in the immediate vicinity. The modified ant algorithm does not recognize ant mills either, but reacts only to the ant density in the ants vicinity. The reason why an ant algorithm can solve a task at all, such as finding tracks or avoiding possible ant mills, lies in the swarm behavior. Collectively a task is solved in the swarm and not each individual separately. E.g. with a strong evaporation of pheromones, a single ant may no longer find its self-laid trail to the food source. Only other ants in contact with this trail reinforce it for such a long time that a complete connection between nest and food can be maintained.

We now describe our algorithm 1 in detail and explain it using the figures 1 and 2 as examples. We have omitted data such as the distance before the ant's death, the counting of age and the logic of starvation from the algorithm, as this is for evaluation purposes only. In our figures we use orange color and a number for a grid location with food and nutritive value and green as a pheromone with a number, that represents its potency. In addition to this figures, we have listed important internal variables of the ant object and the algorithm in table I to provide a quick overview.

Initially, all ants in the nest start with $mode$ ="search". Our modification has the consequence that all ants start an

Algorithm 1 the modified ant algorithm

```
1:  procedure WAKEUP
2:      if escapeSteps > 0 then                          ▷ escape behavior
3:          go 1 step forward
4:          escapeSteps ← escapeSteps − 1
5:          if escapeSteps = 0 then
6:              mode ← "search"
7:          end if
8:          return
9:      end if
10:     if mode = "homing" and I am at the nest then
11:         mode ← "search"
12:     end if
13:     potency ← potencies of pheromones around me
14:     count ← count neighbor ants
15:     if I am on food then
16:         nu_last ← nutritive from that food
17:         nu_big ← max(nu_last, nu_big)
18:         mode ← "homing"
19:         decrease nutritive from that food
20:     end if
21:     if mode = "homing" then
22:         if I am on a pheromone position then
23:             add 0.2 * nu_last / nu_big to its potency
24:         else if I am on a pheromone free position then
25:             create pheromone
26:             set its potency ← nu_last / nu_big
27:         end if
28:         orient to nest
29:         go 1 step forward
30:         return
31:     else
32:         if count ≥ 3 then                            ▷ start escape behavior
33:             orient to 1 of 6 random directions
34:             escapeSteps ← 5
35:         else if There are pheromones around me then
36:             score_i = potency_i * weight_i
37:             go to the position with biggest score
38:             add 0.1 * max score / Σ score to its potency
39:         else
40:             orient to left, right or straight ahead (random)
41:             go 1 step forward
42:         end if
43:         return
44:     end if
45: end procedure
```

TABLE I
INTERNAL VARIABLES

name	initial	meaning
$mode$	"search"	We store here whether the ant is carrying food or searching for it.
$escapeSteps$	0	It indicates how many more steps the ant has to walk to escape from a traffic jam. This is actually a third mode, but it never occurs in $mode$ = "homing".
$count$	0	number of ant neighbors
weight	$\{2, 3, 1, 0, 1, 3\}$	This array of constants is used to weight the pheromones perceived by the ant. The weighting is motivated by the positions of the ant's antennae, should encourage running forward and an alternative direction to the left or right. We have shown it in figure 1 (b).
potency	$\{\}$	the pheromone potency of the 6 grid fields around the ant clockwise starting at the front side of the ant
score	$\{\}$	results of $potency * weight$
nu_{last}	0	the last nutritional value the ant saw
nu_{big}	1	the greatest nutritional value the ant has ever seen

initial escape behavior. Most of the ant mills we found in our simulation had a diameter of less than 4, so if an ant accidentally runs into an ant mill to escape, it crosses it completely in 5 steps. We thought this was a good idea. If an ant has run this 5 step as escape distance, the "search" mode takes effect again. At this point we are already finished with our change of the classic ant algorithm.

Now, the ant scans all 6 grid fields around it and stores the pheromone strengths relative to its running direction into the $potency$ array. If a field has food instead of pheromone, this is treated separately later. The index $i = 1$ of the $potency$ array is for the field directly in front of it, 2 is for the front right of it, 3 is for the field right behind it... In addition, it counts all the ants there so that it has a basis as to whether she has to flee. If this is not the case and no pheromones or food are found, the ant chooses a random direction, whereby we forbid the fields behind it, as well as behind left and behind right. Our ants should preferably run forward. After the choice of direction and a single step forward, the *wakeup()* procedure of the ant ends and is not executed until the next simulation round.

In the vicinity of food and pheromones the behavior of the ant changes, which is activated with $wakeup()$. Let us first explain the case from figure 1, where around an ant only has to react to 2 pheromones. There is still another ant (gray) shown, but our escape behavior starts only from 3 more ants on, so

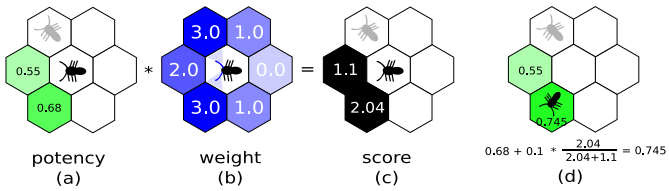


Fig. 1. ant next to pheromones

the other ant can be ignored. The ant now evaluates the fields with the pheromones. It does this with a $weight$ array. These are factors defined by us, which reflect the characteristics of the sensory system and the ant's willingness to make special decisions e.g. running to the left or to the right has a factor by 3.0 and straight forwards only 2.0. The pheromone potency is multiplied by these factors, which depend on the running direction of the ant, and this value is stored in the $score$ array. The ant now goes where the score is highest. It now marks this decision with a pheromone release. The ant can add a maximum of 0.1 to the potency. How much they use of it makes them dependent on the ratio between the score of the selected field and all scores together. In figure 1(d) we have added the calculation as an example.

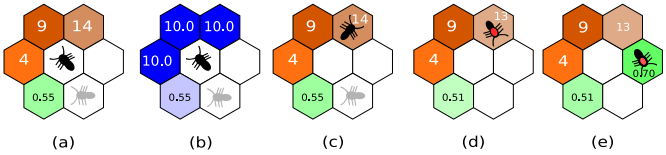


Fig. 2. ant next to food.

In another case, when there is food, the procedure is as shown in figure 2. To prefer food to pheromones, the score of these fields is set to 10. In our algorithm 1 we omitted this exception for simplicity. The ant randomly selects one of the food fields and goes there. At the next $wakeup()$ first the nutritional value of the food is stored in nu_{last} and the largest of these values is stored in nu_{big} by the ant so far. In addition, the nutritive value is counted down. The ant now switches to homing mode, orientates itself towards the nest (figure 2(d)) and takes a step forward. At the next $wakeup()$ in this mode new pheromones are placed or existing ones are adjusted to the ratio nu_{last}/nu_{big}. In figure 2(e) an initial pheromone release with $potency = nu_{last}/nu_{big}$ is generated, because there was nothing there yet. If there had already been a pheromone, we would have added $0.2 * nu_{last}/nu_{big}$ to potency. Additionally, between (c) and (d) the pheromone potency decreased by 6% (evaporation) after a new $wakeup()$ call.

Once the ant has reached its nest, the hunger counter, which was stopped during transport of food, starts again at zero. For simplicity we have not shown this variable in algorithm 1 either.

D. Measurements and parameters

We simulated our 3 different ant algorithms with 10 different worlds with fixed, initial random placement of food

areas. For each algorithm (classic, modified, random) we got an average value by adding the measured results together. We took a focus on these values:

- average age of the ants in the colony during the simulation (and finally)
- reduction of the food in the world during the simulation
- number of dead ants over all 10 worlds
- distance covered of the last 10 steps before the ant dies

An ant's age is the number of simulation rounds it has experienced since birth. As described in subsection III-B5, an ant from the age of 50 dies of starvation if it cannot find food. Each food find increases her age by at least another 50 rounds. The rounds in which it carries the food to the nest are added. By considering the average age we wanted to have a size that is influenced by the food supply, irrespective of the quantity and accessibility of the food sources and (in future simulations) irrespective of the number of simulated ants. If the food supply of the colony is good, the average age increases, which we consider to be successful.

The amount of food that ants have carried away from the world is the second value that is important to us. It also gives a direct indication of how successful our modified algorithm is in collecting food compared to a randomly running ant or the classic ant algorithm. How evenly the task of collecting food is distributed in the colony can be determined by this value too, but we find the average age for more intuitive.

As the last measurement we chose values that describe the death of an ant in more detail. We concentrated here on the last 10 rounds an ant experiences before it dies. With every ant that died, we saved the distance between their place of death and the place 10 rounds earlier. We understand these values as an indicator of mass mortality in a small area (ant mill) or on a trail where there is no more food at the end. In subsection IV we go into more detail. For the representation we formed a histogram (figure 4), in which the deceased ants are assigned to discrete distances. For better readability, we did without a typical bar display of a histogram.

In our measurements we varied besides the three algorithms the evaporation. We had also changed other parameters such as the complete $weight$ array and the strength of the change in pheromone potency, but we found similar results as if we were only changing evaporation. Therefore, we have only varied the evaporation and the other possible parameters we left constant.

IV. Results

In this section we first describe our results and clarify their connections with a view to solving the ant mill problem. For a first overview please have a look at Figure 3. There we have shown in 2 plots the development of the average age and the collected amount of food with an evaporation of 4% over the entire simulation period (100 rounds). In all graphics we have shown the values of all three algorithms to allow a comparison. The sawtooth shape at average age is very noticeable in the classic form of the ant algorithm and in the random version.

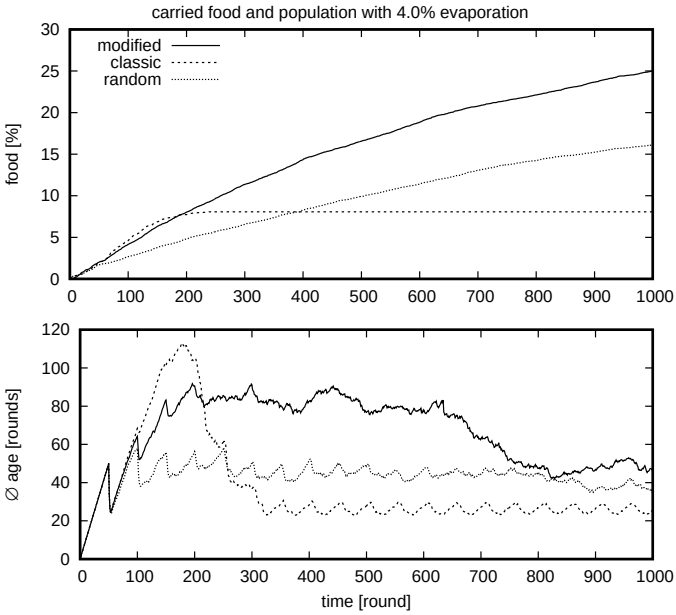


Fig. 3. Cumulative values for average age and carried food of all simulated worlds with an evaporation of 4%.

This occurs when many ants die at the same time, as they are reborn at the age of 0 simulation rounds. We could have prevented this sawtooth pattern by recreating the ants with a random age. In any case, however, we would have lost the possibility of recognizing regular mass deaths at this point. In this measurement curve it is also interesting that in the classic version the average age first increases enormously and then lies significantly below the curve for ant, which work with a random algorithm (i.e. without pheromones).

Due to the reduced average age, obviously more ants die regularly in the classic pheromone controlled version than in the random version without pheromones. In the random version, the sawtooth pattern is also less pronounced over time – the ants no longer die at the same time. If you look at the success of the colony in collecting food, in the classical version the amount of food collected increases massively at the beginning (parallel to the increase of the average age), but suddenly stagnates. In the modified version and the random version, however, food is continuously collected. Although both curves follow a typical saturation function, food is obviously still available, while in the classic version in the same scenario food is apparently no longer reached by ants. In our modified version of ant algorithm, both the amount of food collected and the average age are higher than in the other two versions. It is to be expected that the average age of all versions decreases over time and settle to a value of 25. At some point no more food can be reached by the ants and the ants cannot become older than 50 rounds.

Next we take a look at Figure 4 to get more background information about the dying ants. This graph shows the results with 3 different evaporations. These are actually 3 histograms, but for better readability we have not chosen a typical bar

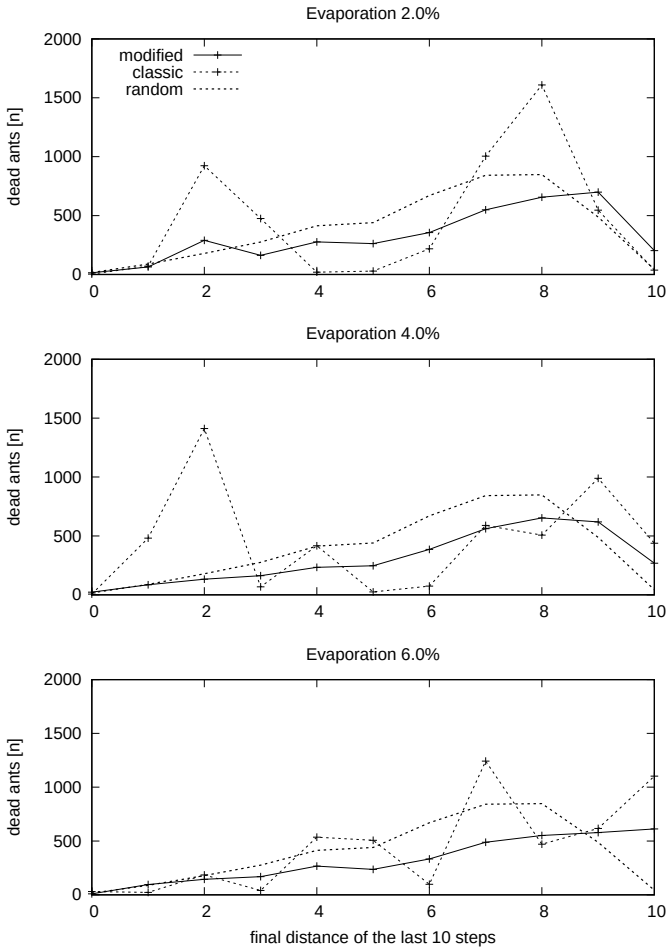


Fig. 4. With an evaporation < 10 the ant algorithm is able to work well, as a marked location takes a bigger effect on the other ants.

graph. In these histograms the values for the random ant algorithm are identical, because this algorithm works without pheromones and therefore evaporation cannot have any influence. As we have already described in section III-D, the histograms show which distance an ant covered shortly before it died. Since ant mills and routes to dried up food sources are characterized by the fact that they cover a particularly short or long distance. This should be reflected by peaks in the histogram of the dead ants, because in that cases these ants do not find any food (especial in ant mills). In these 3 plots we have these peaks in the classic ant algorithm and in our modified version of the algorithm they do not occur. Since the described problems cannot occur with the random ant algorithm, this curve serves as a good comparison of how such a histogram should look without ant mills. What you can also see in the 3 plots is that the number of dead ants (especially the height of the peaks) decreases with a higher evaporation.

Let us now conclude with Figure 5 to take a closer look at the aspect of evaporation. In this graphic, the total average age of the ant colony is plotted against the evaporation with

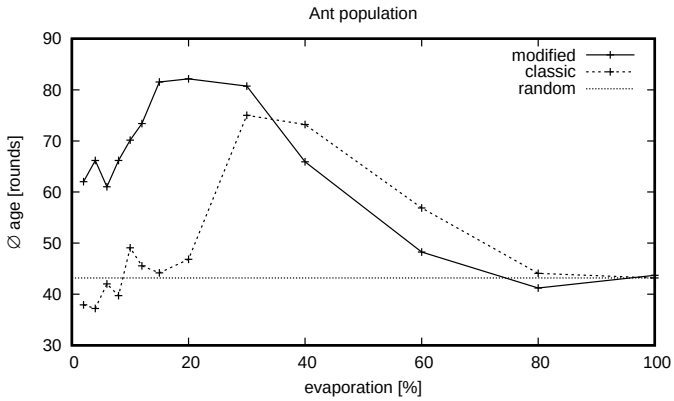


Fig. 5. Average ant age in relation to evaporation

which it was simulated.

Random is entered here as constant, since the algorithm is independent of evaporation. The curves of modified and classic approach this constant with increasing evaporation, since the effect of the trail-laying on the running behavior of ants decreases. Even though "modified" as a whole brings a higher average age to the ant colony, the average age is higher in the classic version after an evaporation of 40%.

A. Discussion

Figure 4 is the most interesting one, because we have big peaks here. We plot the results with and without our modification on different evaporation values and get similar results. Let's look at the middle graphic. These high death rate peaks are our ant mills and high frequented trails to empty food areas. The peaks on a distance at 1 and 2 are ant mills. These more than 1200 moribund ants covered a distance of only 1 or 2 for the last 10 steps before dying. The other high peaks at a distance of 7 and 9 are coursed of trails to empty food locations. These ants are running straight forward the last 10 steps before their fateful death. Our algorithm does not have a high effect on these ant behavior. In "Chaos–order transition in foraging behavior of ants" [13] you can find a solution: after the ant finds food, it starts a small random walk to come into a (for this ant) well known area around the nest location. If the ant is in this area, it directly goes to it. The result in this solution is a big circular trail back to the nest instead of a jammed autobahn to a place without food. Our ants are running back to nest the direct way immediately, thus we still have this problem.

In Figure 5 we detect with additional jam and evaporation values, that there is a "early" hill with a evaporation value below 40%. We think in our configuration there is a well defined combination of the number of ants, the partial addition of pheromones, the maximal distance between food areas and the evaporation value. While our modified ant algorithm, which is less dependent on evaporation, increases the average age of ants, which can be attributed to a better food supply, the classical algorithm produces ant mills when evaporation is unfavorably low.

We think it is very important that an unfavorable evaporation does not lead to a malfunction of the ant algorithm. If you use the classic ant algorithm on robots on a distant planet, where certain environmental factors regarding atmosphere or dust are difficult to simulate, our modified version of ant algorithm would be the better choice. If we look at Figure 4 again, we have to realize that our modified version has no peaks with high mortality rates and the values are very close to those of the ants running the random ant algorithm. In Figure 3 you can also see that the amount of food collected is significantly higher. We consider both aspects a complete success.

V. Conclusion

In this paper we investigated the occurrence and resolving of ant mill issues in ant-based algorithms. Our modified ant algorithm with a kind of traffic jam control is a solution to prevent ant mills and makes collecting food in a swarm more successful. We show that our algorithm reduce the rate of death ants, which are running only in a small area before dying. This has a big effect on the average age in the colony, as the ants find more food instead of running in small circles and dies.

For future work, we have further going ideas. At the end of our work we varied the algorithm a little. For example instead of marking the pheromone tile where the ant is going to, we copy its potency to the actual location and modify the potency on that location. That was not a classic ant algorithm, but it still worked. We varied our $weight$ array too, that the ant uses, with only 1.0 in it. The results changed very little from that. The $weight$ array is a relict of the other ant algorithm we found, were a favorite ant location is based on known food locations in a linear optimization process [11]. We think, that we can simplify this part in our algorithm and can make it more lightweight. We also dealt with the simulation of bees (see [15], [16]) and planned to compare this concept with that of the army ant. We also found the idea that a nest in nature is moved by the army ant and produces a new breed very interesting. We could make this "restart" e.g. depending on the amount of food collected. In addition, there are other ants in nature, not as uniform as the army ant. Here we would be interested to know how they manage to prevent misbehavior in the swarm through different "behavior anomalies of single individuals".

We read that pheromones are not limited to one species [17]. For example, we could simulate what happens to a malignant beetle that leaves "false pheromone trails" and how to react to them in the swarm. We also asked ourselves how strongly the relationship between the number of ants, their hunger counter and the width of the pheromone traces plays a role.

Our ant algorithm is a *local algorithm* [18]. It is distributed (reacting on pheromone "data" place by other participants), running in constant time and it is independent of the size of the network (number of ants). Running in constant time: We use a sorting algorithm to select the biggest scored grid as new location. But getting the maximum value in a set 6 values (hex grid!) is not really complicated. Our findings on secure ant algorithms in [19]–[21] can be further used to analyze the problem of the emergence of the ant mills.

Acknowledgments

We would like to thank the Arab-German Young Academy of Sciences and Humanities for funding this research as well as our collaboration partner Dr. Ahmed Khalil for the valuable discussions on future nanorobot technologies.

References

[1] M. S. Blum, J. Roberts, and A. Novak, "Chemical and biological characterization of venom of the ant solenopsis xyloni mccook," *Psyche: A Journal of Entomology*, vol. 68, no. 2-3, 1961.

[2] D. Fletcher and M. Blum, "Pheromonal control of dealation and oogenesis in virgin queen fire ants." *Science (New York, NY)*, vol. 212, no. 4490, 1981.

[3] J. Tentschert, H.-J. Bestmann, B. Hölldobler, and J. Heinze, "2,3-dimethyl-5-(2-methylpropyl)pyrazine, a trail pheromone component of eutetramorium mocquerysi emery (1899) (hymenoptera: Formicidae)," *Naturwissenschaften*, vol. 87, no. 8, 2000.

[4] T. C. Schneirla, *A unique case of circular milling in ants, considered in relation to trail following and the general problem of orientation*. American Museum of Natural History, 1944. [Online]. Available: http://hdl.handle.net/2246/3733

[5] F. Delsuc, "Army ants trapped by their evolutionary history," *PLOS Biology*, vol. 1, no. 2, 2003.

[6] D. C. Magri, G. J. Brown, G. D. McClean, and A. P. de Silva, "Communicating chemical congregation: A molecular and logic gate with three chemical inputs as a "lab-on-a-molecule" prototype," *The American Chemical Society*, vol. 128, no. 15, 2006.

[7] A. Prasanna de Silva and N. D. McClenaghan, "Proof-of-principle of molecular-scale arithmetic," *Journal of the American Chemical Society*, vol. 122, no. 16, 2000.

[8] S. Kassem, A. T. Lee, D. A. Leigh, V. Marcos, L. I. Palmer, and S. Pisano, "Stereodivergent synthesis with a programmable molecular machine," *Nature*, vol. 549, no. 7672, 2017.

[9] V. Balzani, A. Credi, F. M. Raymo, and J. F. Stoddart, "Artificial molecular machines," *Angewandte Chemie International Edition*, vol. 39, no. 19, 2000.

[10] R. A. L. Jones, "Soft machines: nanotechnology and life," 2004.

[11] A. Colorni, M. Dorigo, V. Maniezzo *et al.*, "Distributed optimization by ant colonies," vol. 142, 1991.

[12] I. D. Couzin and N. R. Franks, "Self-organized lane formation and optimized traffic flow in army ants," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 270, no. 1511, 2003. [Online]. Available: http://rspb.royalsocietypublishing.org/content/270/1511/139

[13] L. Li, H. Peng, J. Kurths, Y. Yang, and H. J. Schellnhuber, "Chaos–order transition in foraging behavior of ants," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, 2014.

[14] S. Sommer, C. von Beeren, and R. Wehner, "Multiroute memories in desert ants," *Proceedings of the National Academy of Sciences*, vol. 105, no. 1, 2008.

[15] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "-the bees algorithm—a novel tool for complex optimisation problems," 2006.

[16] Z. S. Liang, T. Nguyen, H. R. Mattila, S. L. Rodriguez-Zas, T. D. Seeley, and G. E. Robinson, "Molecular determinants of scouting behavior in honey bees," *Science*, vol. 335, no. 6073, 2012.

[17] F. Gonzalez, "Smells of sociality," 2017.

[18] J. Suomela, "Survey of local algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, 2013. [Online]. Available: https://users.ics.aalto.fi/suomela/doc/local-survey.pdf

[19] P. Mogre, K. Graffi, M. Hollick, and R. Steinmetz, "AntSec, WatchAnt, and AntRep: Innovative Security Mechanisms for Wireless Mesh Networks," 2007.

[20] K. Graffi, P. S. Mogre, M. Hollick, and R. Steinmetz, "Detection of colluding misbehaving nodes in mobile ad hoc and wireless mesh networks," in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, 2007, pp. 5097–5101.

[21] P. S. Mogre, K. Graffi, M. Hollick, and R. Steinmetz, "A security framework for wireless mesh networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 3, pp. 371–391, 2011.