Lightweight Pheromone based search algorithm with Ant Mill protection

Jochen Peters
Heinrich-Heine-Universität Düsseldorf
Institut für Informatik
40225 Düsseldorf, Germany
Email: jochen.peters@hhu.de

***Abstract*—The technological development is often motivated by nature concepts. For example molecular machines with a minimal logic acting like complex biological proteins. We are thinking ahead: these molecular machines should work as a robot swarm in the future. Therefore we need a simple logic for each machine without traditionally wireless network communication. We concentrated on the ant algorithm to find and collect food. This algorithm use pheromones and the environment to control and communicate to each machine (ant). We found out, that simulating a simple implementation of this algorithm leads to ant mills and high frequented routes to empty food areas. In this paper we present a small modification of the ant algorithm to protect a swarm from collective misconduct. We show, how successful this modification is and compare this concept to nature and other ant algorithm simulations.**

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

I. Introduction (not ready)

Raketensteuerungen von Atombomben werden oft von pneumatischen (Staubsaugermotoren und Klappen wie bei Zuse I) oder hydraulischen Systemen betrieben, damit sie bei einem elektromagnitischen Impuls einer anderen Bombe noch weiter funktionieren. In der Atmospähre eines anderen Planeten könnte eine Elektronik z.B. wegen der starken statischen Ladung des Marsstaubs ähnliche Probleme verursachen. Unter Wasser oder in einem sandigem Medium können GPS und Funkwellen kaum bis gar nicht genutzt werden. Schaut man sich aktuelle Nanotechnologien an, so werden chemische Botenstoffe zur Auslösung bestimmter Reaktionen, interessanter, als mit Funktechnik Roboter in unserer Blutbahn kommunizieren zu lassen. In der Natur werden bereits Hormone (und keine Funkwellen) als Kommunikation eingesetzt. Sollte man chemische Propeller an Makromolekülen bilden oder betreiben, die durch eine chemische Reaktion aufgrund einer bestimmten Konzentration eines künstlichen Hormons an der Stelle des Moleküls, an dem der Propeller sitzt, so sind logische Operationen oder gar Speichermedien in einem Molekül begrenzt. Durch künstliche Methabolismen eine turingvollständige Sprache bilden, die als Eingabe Konzentrationen von bestimmten Molekülen akzeptiert und gleiches als Ausgabemedium zur Steuerung einer molekularen Struktur erzeugt, mag heute noch nach Science Fiction klingen - findet aber bereits in Laboren statt. (todo: hier fehlt mir noch eine Referenz)

In contrast to bees, the simulated ants did not split into a searching and a collecting behavior in that case.

We added a simple traffic jam control to each unit and measured a better food supply.

This demo file is intended [1] to serve as demo. I wish you the best [2] of success. test test [3] ...

Molecular machines, optical digital processing units and analogue computers with a pneumatic logical unit are able to work in special environments, where modern micro controllers are too large, use too much electric energy or are not resistant to massive electromagnetic forces. Many of these techniques are unable to use wireless communication, unable to store big data sizes or do millions of instructions of a complex algorithm. If you want to solve a big problem with it, you have to solve it in a collective opportunistic way with a simple logic for each unit.

Primitive Algorithmen wie den Ant-Algorithmus sind bereits seit vielen Jahren bekannt. Dieser nutzt einen Duftstoff, der in einer Region von jedem Teilnehmer freigesetzt wird, um eine Art kollektiven Speicher zu verwirklichen. Ziel ist es auf diese Art Spuren für andere Teilnehmer zu hinterlassen, welche zu Futterquellen führen. Dieses Futter soll dann ins Nest getragen werden. Je mehr Ameisen auf einer gelegten Spur laufen, desto stärker wird diese Spur. Theoretische Modelle zeigen, dass sich dadurch sogar kürzere Wege zurück ins Nest durchsetzen, da diese Wege von den Ameisen, die ihn gefunden haben, häufiger frequentiert wird - die Duftspur wird so verstärkt. In der Literatur finden sich viele Variationen und theoretische Modelle, wie ein solcher Algorithmus, durch die Natur motiviert, nachgebildet werden kann. Z.B. sind lokale Algorithmen (verteilte Algorithmen mit linearer Laufzeit, die nicht von der Größe des Netzes abhängig sind), einem Schwarm-Algorithmus wie dem Ant-Algorithmus sehr ähnlich, betrachten aber keine kontinuierliche Veränderung der Daten Aufgrund von Bewegungen. Schaut man sich in der Literatur die Ant-Algorithmen im Detail an, so wird der Erfolg einer Ameisen-Kolonie bei gewählten Parametern, die das Legen der Duftspuren und deren "Verwitterung" beeinflussen, kaum behandelt.

II. Related Work

We searched for a very common ant algorithm and we did not find *the* ant algorithm.

The first ant algorithm is documented at [4] in 1991 and they try to find a heuristic to solve the traveling salesperson problem (TSP).

...

These points are very common in an ant algorithm:

- if ant found food, it runs back to home (*homing*)
- ant "looks" around
- if ant found a high potency pheromone location, it makes it more "dusty" with a additional pheromone secret
- pheromones lost their potency by time
- ant does a random walk (but not going backwards) or goes to the high potency pheromone location
- ant makes only one "move" after a decision

These points may differ in every ant algorithm we found:

- let single ants run on a line many times
- initial position and initial potency of pheromone tiles
- initial ant positions
- the concept, how an ant running back to home
- ant speed
- setting or not setting pheromone tiles all the time
- using or not using a blur effect on pheromone tiles
- a concept for born and death of an ant
- the concept how the ant makes their decision (the "looking around")
- objects between food tiles and home location

One of the basic problems is, how the ant makes their decision. We start our simulation *without initial pheromone trails* and *our ants have no eyes* or getting information about food locations. In the publications we study, they main idea was to find a shortest way to known food positions. Their ants made a decision on that information.

Until now the ant algorithm is used to find short trails to a single food location in a scenario with given initial pheromone trails. In these scenarios ant mills and high frequented routes to empty food areas are less possible. Their simulations stops after the food area is empty or (because they focused only on the building of trails) the food area is not decremented. We found a focus on simulating ant mills in [5] but not with a combination of building initial pheromone trails by the ants or searching for food.

III. Methods

The simulation of our ant algorithm was splitted into the following parts:

- a simulator (doing 1000 loops)
- a world (2 dimensional)
- food tiles (marked as food with a nutritive value)
- pheromone tiles
- 30 robot objects (each with our ant algorithm)

Additionally to the normal values we need in the simulation, we logged the *mean of the ant age* and the *last 10 steps before dying*.

In this chapter we explain each part and who we initialized it with fixed values. The last chapter is about our ant algorithm and the traffic jam control to prevent ant mills.

A. Simulated parts and their interaction

We did not want to print any code here to make it easy to implement the parts, concept and objects in your preferred system, language or medium.

1) World: Our world is a 2 dimensional hex grid. Thus our robots and tiles can only placed into that grid. The worlds are generated with 10 different random seeds to place food tiles 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 each robot and tile must fit into the hex grid. If we talk about distances we use the euclidean distance based on that x and y values.

2) Robots (ant): We initialized the simulator with 30 ants placed on $(0, 0)$ in the world (we call it *home*). Thus they start with a traffic jam. Each ant has a *hunger level*. If an ant does not reach any food or carries it to home, the hunger level is increment by one. If the limit reaches **50**, it dies immediately and respawns at home. Every ant is awaked by each simulator iteration and runs our ant algorithm with traffic jam control.

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 awaked 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 tiles: Every food tile is part of an food area. In a world generated by a seed these areas can have a size between 1 and 16 tiles. A world can have up to 30 areas. Each tile is generated with a nutritive value between 1 and 20. If an ant reaches the food tile, the nutritive value is decremented by one. If the nutritive value is zero, the food tile 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
- ordered

The spray concept setting an area on a random position and place 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 home location. The position is vary by 5.

We decided to choose these concept to get random worlds with reachable food for the ants. Vary the maximal distance of the ant and/or vary the maximal distance of each area may focused in future work.

4) Pheromone tiles: The pheromone tiles are generated by ants, if an ant finds food and running back to home. This is the only way creating tiles. We store the pheromone potency internally in an alpha value with 1.0 as maximum. The ants are able to increment this value. A tile with a *potency* less than 0.0001 will be removed by the simulator. You can define it as a detection level of an ant, if you want.

After all ants do their work after a simulator iteration, the tiles will be deleted or the potency will be reduced by a *evaporation*:

$$potency_{i+1} = \frac{100 - evaporation}{100} * potency_i$$

5) Simulator: The Simulator loads the world with it special seed and iterated 1000 times through 30 ants. Thus a single ant is being $wakeup()$ for 1000 times to walk, detect, collect or setting/changing a pheromone tile.

6) Ant age & last steps: In order to be able to establish a statement about the success of all simulated worlds with variation of selected parameters, we decided to log the mean of the ant age after each simulation iteration. To detect a bad behavior like ant mills and high frequented routes to empty food areas we logged the last 10 steps of the ant before its death. Counting the revisited locations and measure the running distance let us understand, what happened to the ant (running in circles or continuous forward).

The age of an ant is the number of "wakeups" from the simulator. Normally an ant will die without food after 50 wakeups. If an ant finds 1 nutritive, it sets its *hunger level back to zero* and the ant lives longer. Thus a high average value of the ant age is a good indicator, that the ant colony finds continuously food and is not running in circles all the time or being on a dreadful track.

B. Traffic jam controlled algorithm

We used a score array to prefer special directions based on the orientation of the ant. This was motivated by the following aspects:

- sensor placement
- prefer running forward
- prefer alternative sideways instead of the main (maybe not optimal) trail

The traffic jam control based on a single *JAM* parameter. We set it at as a constant on each simulation. We focused on modifying this value and who it takes effect on collected food and the mean age of the ant colony.

Every ant follows this simple logic, after the simulator wakes it up. Initial the ant is in *search* mode and set to position $(0, 0)$ = *home*:

- I am in traffic *jam* mode
  - running 1 step forward
  - decrements my traffic step value
  - if traffic step value = 0: switch to search mode
  - **SLEEP** until Wake up
- **scan and modify**
  - getting potency of pheromone tiles around me
  - count ants around me
  - I am on a pheromone tile:
    - modify pheromone potency based on score
  - I am on a food tile:
    - get nutritive value and store it, if it is bigger than my last stored one
    - switch to *homing* mode
  - I am not on a tile, but I am in *homing* mode:
    - create pheromone tile
    - modify pheromone potency based on score
- I am in *homing* mode and I am at home:
  - switch to *search* mode
- **move**
  - I am in *homing* mode:
    - orient to $(0, 0)$ and go forward
    - **SLEEP** until Wake up
  - there are $\geq$ JAM ants around me:
    - set a random orientation
    - running 1 step forward
    - set traffic step value = 5 -1
    - switch to traffic *jam* mode
    - **SLEEP** until Wake up
  - There are no pheromone or food tiles around me:
    - set a random orientation
    - running 1 step forward
    - **SLEEP** until Wake up
  - Choose a new orientation based on the highest scored tiles around me
  - running 1 step forward

The modification of the pheromone potency of a pheromone tile is calculated by this formulas:

Ant is in *homing* mode:

Ant is not in *homing* mode:

Our random orientation is modified a bit to permit running backwards. Only running forward, left or right is allowed.

IV. Results

intro about selected data, getting them and how we analyse them

A. Definitions and Taxonomy

Parameters

- different food area size
- different distance between food area and home
- different count of food areas
- different direction random size
- different prefer pattern for dusty tiles
- different amount, how strong the way decisions effects on setting the new dust
- better to add or multiply dust on a tile
- loosing dust on tiles rate
- different hunger limits (?)
- different random seeds (?)

Effects

- development of collected food (%) in 1000 iterations
- used dust (sum) over 1000 iterations
- how many died (how long did they life)
- development of random-, jam- and dust-decisions (%) in a simulation
- distance after last 10 steps before dying (ant traps?)

B. Aspect 1

Subsection text here.

C. Aspect 2

Subsection text here.

D. Discussion

intro, offer explanation and reference to literature

V. Conclusion

Our ant algorithm is a *local algorithm* [6]. It is distributed (reacting on pheromone "data" place by other participants), running in constant time [1] and it is independent of the size of the network (number of ants).

Ebenso gibt es beim Ant-Algorithmus einen Aspekt, der bei der ältesten Art, den Wanderameisen, besondere Aufmerksamkeit verdient: "Ameisenmülen". In diesem Fall laufen alle Ameisen im Kreis und sterben an Erschöpfung, da sie so keine weiteren Futerquellen finden. Ein weiterer Punkt, der einen Erfolg eines zu primitiven Ant-Algorithmus in Frage stellt, sind Ameisen-Straßen, die zu versiegten Futterquellen führen. Wir konnten in unseren Simulationen beide "Fehlentwicklungen" beobachten.

Man kann den Ant-Algorithmus durch Hinzunahme von GPS, Funktechnik und einem großen Arbeitsspeicher zur modifizieren, um ein solches Fehlverhalten zu erkennen oder zu verhindern, aber diese Modifikationen setzen folgendes voraus:

- elektrische Energie
- Rechenleistung
- mindestens die Größe des Chips
- bedenkenlose Anwendung von Funkwellen
- Speichermedium

Es ist also nicht vollkommen abwegig, einen Ant- Algorithmus so simple wie möglich zu halten, der die Umgebung als Massenspeicher nutzt, welche mit dem Operationsgebiet mitwächst. Wir haben diesen Aspekt der Einfachheit in Hinblick auf Erfolg und Robustheit bzgl. äußerer Parameter untersucht und kommen zu dem Ergebnis, dass auch andere natürliche Aspekte der Wanderameise eine wichtige Rolle Spielen müssen, was diese Spezies so lange parallel zu anderen Ameisenarten hat erfolgreich koexistieren lassen.

Future Work: new open questions? how can we find answers in the future? How can we use our solutions in the future?

References

[1] 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, pp. 377–380, Aug 2000. [Online]. Available: https://doi.org/10.1007/s001140050745

[2] 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, pp. 8392–8397, 2014.

[3] F. Gonzalez, "Smells of sociality," Ph.D. dissertation, 2017.

[4] A. Colorni, M. Dorigo, V. Maniezzo *et al.*, "Distributed optimization by ant colonies," vol. 142, pp. 134–142, 1991.

[5] 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, pp. 139–146, 2003. [Online]. Available: http://rspb.royalsocietypublishing.org/content/270/1511/139

[6] J. Suomela, "Survey of local algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 24, 2013.

[1] Running in constant time is not really true. We use a sorting algorithm to select the best pheromone tile as new location. But getting the maximum value in a set 6 values (hex grid!) is not really complicated.