Ant Colony

Clustering & Sorting

Magnus Erik Hvass Pedersen (971055)
Daimi, University of Aarhus, July 2003

1 Introduction

The purpose of this document and accompanying software is to verify attendance of the author to the *Swarm Intelligence* course at DAIMI, University of Aarhus.

First the basic ant colony clustering and sorting models are outlined, then a number of alterations are reviewed and others proposed, and finally the implementation is described.

The reader is assumed to be familiar with swarm intelligence and related topics.

2 (Corpse) Clustering

Ants exhibit many kinds of behaviour that can be modelled in computer algorithms for solving problems. An example is corpse clustering in which an unladen ant will pick up a lone corpse of a fellow ant and drop it where there are more corpses to form a cemetery. In modelling, the picking up and dropping is decided by a probabilistic measure on the density of corpses in the vicinity of the ant. The wandering in its simplest form is completely random.

When corpses are placed in a discrete arena (grid) with a maximum of one corpse per cell, the density $f \in [0, 1]$ is simply the fraction of neighbouring cells containing a corpse. The probability p_p that an ant will pick up a corpse is then:

$$p_p = \left(\frac{k_p}{k_p + f}\right)^2 \tag{1}$$

Whereas the probability p_d that an ant will drop a corpse it is carrying, is given by:

$$p_d = \left(\frac{f}{k_d + f}\right)^2 \tag{2}$$

Where k_p and k_d are akin to activitation thresholds on f.

3 (Brood) Sorting

Ants also perform brood sorting motivated by different needs for different life stages of the brood. The clustering model is extended so that the density of brood reflects their similarity also, causing them to be picked up if they are dissimilar to their surroundings and dropped in a cell whose neighbourhood is of similar character. The model that will be the basis of this document is due to Lumer and Faieta and is outlined in [1].

The *local density* or similarity of an object (brood) $o \in O$ to its neighbours is given by:

$$f(o) = \max\left(0, \frac{1}{S} \sum_{o_i \in N_s(l)} \left[1 - \frac{d(o, o_i)}{\alpha}\right]\right) \tag{3}$$

Where $N_s(l)$ is the set of neighbouring brood centered around location l, having at most S elements. When the arena is a grid, a natural choice of $N_s(l)$ is the square of $s \times s$ cells surrounding and including l: $N_{(s\times s)}(l)$, in which case $S = s^2$. In these experiments, a neighbourhood of 5×5 is used. The similarity measure $d(\cdot, \cdot)$ is any valid *metric* on the objects, and since there are finitely many objects in the arena, it can be assumed to be *normalized* to the range $[0, 1]$.

The parameter $\alpha > 0$ is a scaling factor used for magnifying the dissimilarity between objects. This results in $f(o) \in [0, 1]$ with some *saturation* occuring - depending on α and the dissimilarity of the present objects.

The probability for an ant to pick up an object o placed in the arena at location l, is similar to Eq. (1):

$$p_p(o) = \left(\frac{k_p}{k_p + f(o)}\right)^2$$

The probability for the ant to drop an object o when wandering across a vacant location l is different from that of clustering though:

$$p_d(o_i) = \begin{cases} 2f(o) & , f(o) < k_d \\ 1 & , \text{else} \end{cases}$$

Where k_d is a threshold constant, but Eq. (2) was actually found to be more useful, with $k_p = k_d = 0.1$:

$$p_d(o) = \left(\frac{f(o)}{k_d + f(o)}\right)^2$$

3.1 Algorithm

Let $R \in [0, 1]$ be a random number drawn for each use, the basic algorithm is then given by:

- Place brood and ants randomly in arena.
- For a number of steps or until a criterion is met, repeat for each ant:
    - If the ant is unladen and placed on location l occupied by object o, the object is picked up if $R \leq p_p(o)$.
    - If the ant is carrying object o and placed on an empty location l, the object is dropped if $R \leq p_d(o)$.
    - Move the ant to a new randomly selected (neighbouring) location.

3.2 Improvement

A traditional attempt of improving the above scheme is to have ants move at different speeds relative to eachother, coupled with adjustment to their calculation of $f(o)$. The initially smaller clusters are then merged faster - but according to [2] only for quite large speeds (up to the arena radius).

3.2.1 Stagnation Count

If the similarity-function is not evenly distributed, ants may be unable to find locations where objects that are very dissimilar to the others can be dropped, therefore [2] also suggests a *stagnation counter* which decides when the ant finally gets tired of carrying an object and simply drops it. An alternative way of doing this would be to *gradually* increase the probability for an individual ant to drop its object until it is finally 1, regardless of the local density relative to the object.

3.2.2 Average Similarity Scaling

It is further suggested in [2] to also divide the object similarity with their average μ, which is calculated slightly different here:

$$\mu = \frac{\sum_{o_i \in O} \sum_{o_j \in O} d(o_i, o_j)}{2N(N-1)}$$

With $N = |O|$ being the number of objects.

3.2.3 Adaptive Scaling

The similarity scaling α is modified over time in [2], the overall idea being that clustering is sped up if it seems too slow. The clustering activity is measured by ants either picking up or dropping objects. But this is really a matter of balancing, as too much activity means the sorting is probably of low quality. The method used here therefore differs slightly: For every t_α ant-steps, depending on the activity a_α either increase or decrease α:

$$\alpha \leftarrow \begin{cases} \alpha \cdot \alpha_{\text{inc}} & , a_\alpha < a_{\text{inc}} \\ \alpha \cdot \alpha_{\text{dec}} & , a_\alpha > a_{\text{dec}} \\ \alpha & , \text{else} \end{cases}$$

Where $\alpha = 0.5$ initially, the following parameters were used: $\alpha_{\text{inc}} = 1.1$ (although incrementing never occured with these experiments), $\alpha_{\text{dec}} = 0.995$, $t_\alpha = 350$, $a_{\text{inc}} = |A|t_\alpha/1000$, and $a_{\text{dec}} = |A|t_\alpha/50$, and $|A|$ is the number of ants. A more accurate estimate of the gradient on clustering activity and purity could further improve this.

As $\alpha \rightarrow 0$ a single slightly dissimilar object will cause $f(o) = 0$ even though o is surround by otherwise identical objects - this means clusters will be broken up again. Two solutions are proposed, either use boundaries which works well if there are many small clusters or in the extreme case where no two objects are identical: $\alpha \in [0.16, 1]$ (experimental values), or alter the calculation of the local density so that large dissimilarities have less effect:

$$f(o) = \max\left(0, \frac{1}{S} \sum_{o_i \in N_s(l)} \left[\max\left(b, 1 - \frac{d(o, o_i)}{\mu\alpha}\right)\right]\right) \tag{4}$$

Where $b = 0$ is generally safer and also renders the outer max() unnecessary, but e.g. $b = -0.5$ is faster and gives better separation.

3.2.4 Eager Ants

Realizing that ants spend much time walking across empty cells to find objects they may pick up, [2] suggests ants immediately jump to cells that are known to have objects in them. The natural extension of this concept is for ants that carry objects to jump to empty cells in the vicinity of other objects. This means that an ant never considers cells irrelevant to its current desire, be it picking up or dropping. The concept could be further improved by weighting the potential cells, perhaps with some clever sorting of, or perhaps application of *pheromone* to, the datastructures holding the objects and cells of the arena.

4 Brood Perspective

Having only one metric d on the objects to be sorted, means that each ant is able to compare all characteristics of the objects. This is really a super-intelligent ant, and even humans are only able to focus on a few *dimensions* at a time. Thus it makes sense to introduce a *perspective* on the brood so that different ants compare different parts of the objects.

Experiments were carried out with coloured brood (varying both red, green, and blue) and ants being hardcoded to comparing only one of these colours. New equations were also tried out. The local density is still very similar to Eq. (3):

$$f_b(o) = \frac{\beta}{S} \sum_{o_j \in N_s(l)} [1 - \Phi(d_b(o, o_j))]$$

Where the abstract *skewing* (instead of scaling) function is chosen to be $\Phi(x) = x^{\alpha_b}$ so that $f_b(o) \in [0, \beta]$, and $d_b(\cdot, \cdot)$ is the metric used by the given ant. The probability for picking up brood is:

$$p_{pb}(o) = 1 - \min\left(1, k_{p1}(f_b(o))^{k_{p2}}\right)$$

And the counterpart for dropping:

$$p_{db}(o) = \min\left(1, (f_b(o))^{k_{db}}\right)$$

The following parameters have been found to work well: $s = 3 \Rightarrow S = 9$, $\beta = 9$, $\alpha_b = 0.25$, $k_{p1} = 0.15$, $k_{p2} = 1.85$, $k_{db} = 256$, and the neighbourhood is 3×3.

These provide good visual clustering of a regular distribution of colour-intensities, with large (possibly one big) clusters being *rainbow-coloured*. But there is a tendency to lock onto suboptimal solutions when different distributions of colour-intensities are used. With these parameters, the power functions are quite agressive, so one idea would be that they need to self-adapt to the distribution at hand.

5 Implementation

There are three main classes used for modelling brood sorting: Arena (`LArena`), brood (`LBrood`), and ant (`LAnt`). A sub-class should be made of `LBrood`, implementing the particular brood features and the metric, and if brood perspective is desired, sub-classes of `LAnt` may be used to implement these.

The arena uses two helper-classes, specifically the contents of a cell in the arena (`LArenaCell`), and a cell position (`LArenaCellPos`). The grid consists of a two-dimensional array of `LArenaCell`, and the cell in turn contains a reference to the brood that is placed in it, along with some other useful data.

5.1 Eager Pick-Up Ants

The arena has an array of all the brood, both that presently being carried by ants and that placed in cells of the arena. An ant is assigned an element in this array designating the brood it carries. Elements are then swapped when the ant has dropped the brood it carries and subsequently picks up another. This way the first section of the array always only contains brood placed in the arena, whereas the second part is either brood being carried by ants or brood just dropped. Finding brood to pick up is then simply a matter of considering the elements in the first part of the array, hence an ant never wanders across empty cells when searching for brood to pick up. Also, it is not possible for an ant to pick up brood it has just dropped.

5.2 Eager Drop Ants

Since the objective is to cluster similar objects, they should eventually be placed in contiguous cells. An ant searching for a cell to put its brood in should therefore arguably only consider free cells adjacent to cells containing brood. To efficiently implement this, each cell has a count of how much brood it has for neighbours (i.e. $N_{(3\times3)}$), when brood is dropped or picked-up, these counts are updated and the relevant cells are added to or removed from an array of *free neighbours*. The algorithm for doing this has time complexity $O(1)$ (as does the *eager pick-up ant* algorithm above) - but all 8 neighbouring cells are considered upon each drop or pick-up.

5.3 Arena Shape

The arena determines which neighbour-cells are valid, so different shapes are supported: Toroidal, rectangular, circular, or any other. Since the ants only jump to cells that are implicitly valid, this scheme means that ants have no explicit knowledge of how the arena looks.

5.4 Random Generator

A *quick-and-dirty* random number generator from [3] is used. It does feature some *sequential correlation* which is visible if the output is scaled and discretized (e.g. to $0, \ldots, 255$) without any *dithering* applied. The seeds are constant parameters allowing experiments to be repeated.

5.5 Source Code

Implementation is carried out in *C++* using *Metrowerks Codewarrior* on a Macintosh PowerPC. The project-file is `ant.mcp` and it compiles to the executable file `ant`. Parameter-constants may be adjusted in the individual source-files, including `main.cpp` and `LAnt.cpp`. The file `prefix.h` may be altered for easy switching between the normal and the brood perspective models. For visual demonstration the classic *Silly Balls* example has been modified.

5.6 Main Processing

In a more general context the main processing algorithm should be implemented in its own class, but for brevity it has been implemented in `main.cpp` along with the entry-point (`main()`-function).

6 Observations

No comparison of brood sorting algorithms is included, since they are all subject to parameter tuning to some degree. Algorithm alterations such as *eager ants* - particularly for dropping - further cloud iteration comparison with e.g. pheromone based ant-movement as that used in [4]. A couple of example runs are presented in appendices A and B though.

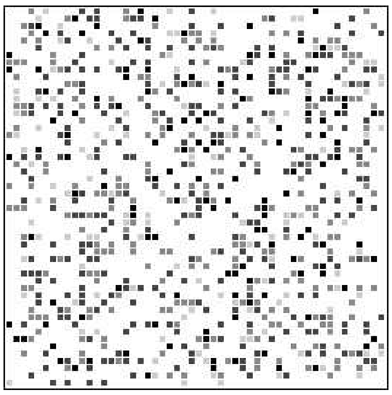
7 Conclusion

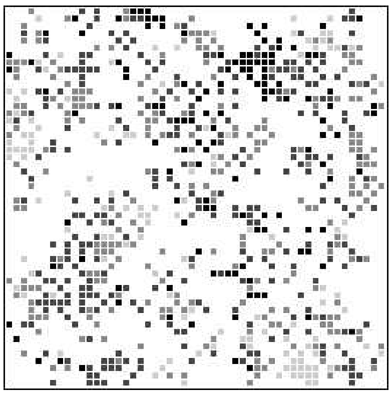
A model for brood sorting in ant colonies was successfully implemented along with several improvements, incl. the (possibly new) concept of *eager drop ants*. Other alterations (e.g. to *adaptive scaling*) were also realized and others merely suggested. The modified algorithms still leave room for improvement though.

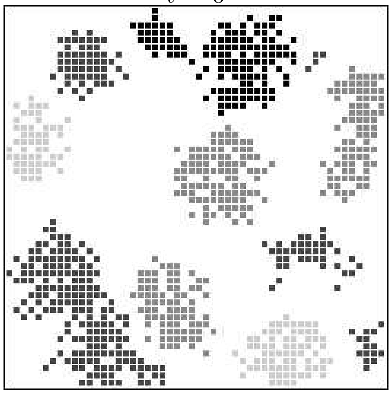
An idea for *brood perspective* in the Lumer & Faieta model was briefly outlined along with experimental alterations to the local density and probability functions.

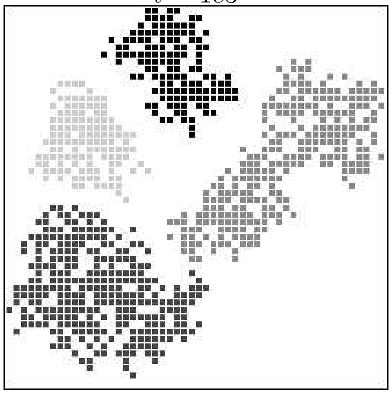
A Example - Grey Brood

For this experiment 800 grey brood objects were placed in a 52×52 cell arena of square topology (i.e. not toroidal). This was sorted by 10 ants, using Eq. (4) with $b = -1$. There are 4 different shades of grey, and these are successfully clustered and sorted. The time-measure t designates iterations for *all* ants to process one step, so the total number of actual ant-steps would be $t_{\text{total}} = t|A|$. All other parameters were as described above.


$t = 0$

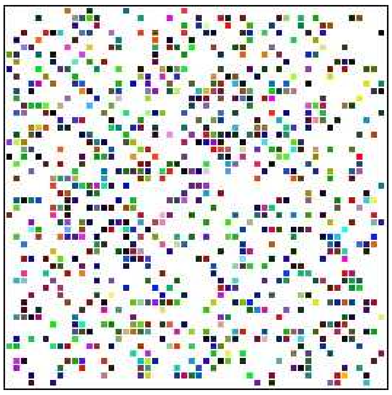

$t = 1e3$

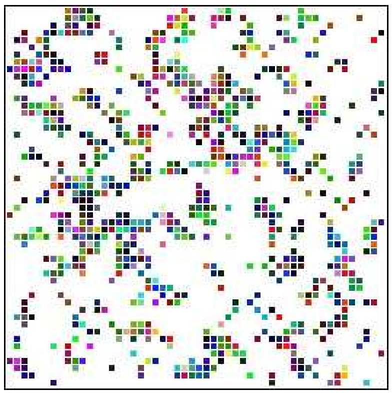

$t = 1e4$

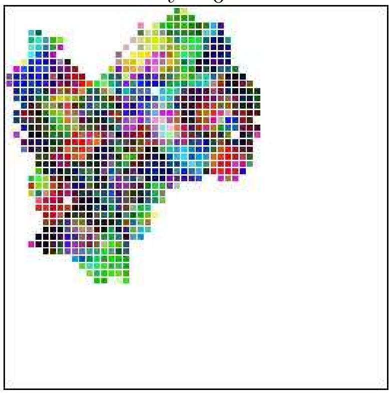

$t = 2e5$

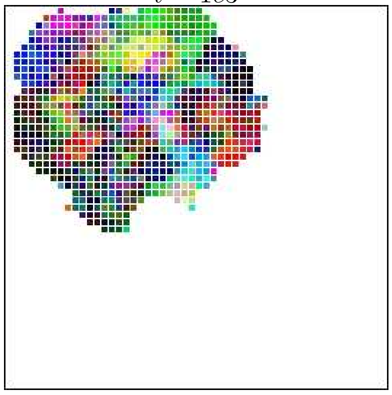
B Example - RGB Brood

Again 800 rgb brood objects were placed in a 52×52 cell arena of square topology. This was sorted by a total of 9 ants, 3 for each colour. Each brood object has a random combination of colours, so no two objects are exactly identical (within correlation bounds). All other parameters were as described above.


$t = 0$


$t = 1e3$


$t = 7e4$


$t = 1e6$

References

[1] Swarm Intelligence
From Natural to Artificial Systems
Eric Bonabeau, Marco Dorigo, Guy Theraulaz
Oxford University Press 1999
ISBN 0-19-513159-2

[2] Improved Ant-based Clustering and Sorting in a
Document Retrieval Interface
Julia Handl, Bernd Meyer
FB Informatik, Unviersity of Erlangen-Nurnberg
School of Computer Science, Monash University, Australia

[3] Numerical Recipes in C
Cambridge University Press 1992
ISBN 0-521-43108-5

[4] Self-Organized Data and Image Retrieval as a
Consequence of Inter-Dynamic Synergistic Relationships in
Artificial Ant Colonies
Vitorino Ramos, Fernando Muge, Pedro Pina
CVRM - GeoSystems Centre, Tech. Univ. of Lisbon (IST)