Ant Algorithms: Part 2

Ant Algorithms

In today's lecture we are going to cover:

- Cemetery Organization and Brood Care
  - For effective clustering
- Division of labour
  - Using the concept of stigmergy to guide task allocation and deallocation.
- Continuous ant colony optimization

Cemetery Organization

Many ant species, exhibit the behavior of clustering corpses to form cemeteries

- Each ant seems to behave individually, moving randomly in space while picking up or depositing corpses.
- The decision to pick up or drop a corpse is based on local information of the ant's current position.
- This very simple behavior of individual ants results in the emergence of a more complex behavior of cluster formation.

Brood Care

A similar type of clustering behavior is observed in some ant species when it comes to brood care.

- Larvae are sorted in such a way that different brood stages are arranged in concentric rings.
- Smaller larvae are located in the center, with larger larvae on the periphery.
- The concentric clusters are organized in such a way that small larvae receive little individual space, while large larvae receive more space.

Cemetery Organization and Brood Care

While these behaviors are still not fully understood, a number of studies have resulted in mathematical models to simulate the clustering and sorting behaviors.

- Based on these simulations, algorithms have been implemented to
  - cluster data.
  - draw graphs.
  - develop robot swarms with the ability to sort objects

Basic Ant Colony Clustering Model

Deneubourg et al. [200] developed a model to describe the simple clustering behavior of ants, for a **single** type of item

- Assume an area is divided into a grid like structure
- The approach depends on the two control parameters: γ_1, γ_2. While not normally named
    - γ_1 can be seen as the desire of an ant to pick up items
    - γ_2 can be seen as the desire of an ant to not drop items
- λ is a value that represents the density of items around an ant (based on the last T steps)
    - Each ant keeps track of the last T time steps, and λ is simply the number of items observed during these T time steps, divided by the largest number of items that can be observed during the last T time steps.
    - You can define the field of view in a couple of ways.

Basic Ant Colony Clustering Model

- An unencumbered ant will pick up an item in it's current position (assuming one is present), with the probability:

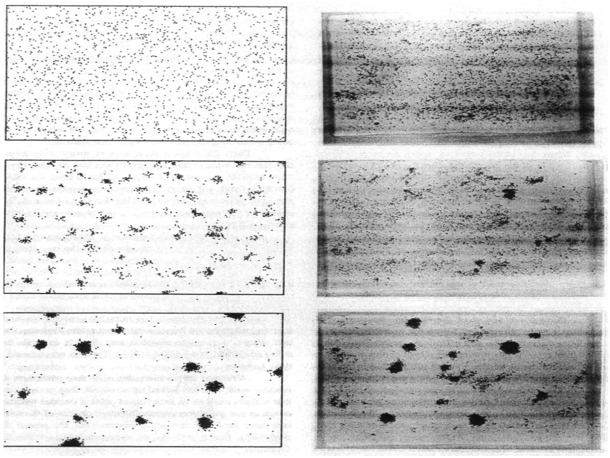
$$P_{pickup} = \left(\frac{\gamma_1}{\gamma_1 + \lambda}\right)^2 \tag{1}$$

  - if λ is much greater than γ_1 the likelihood of of picking up an item is low.
    - area is dense with items so that the ant doesn't want to unclustered
- If an ant is currently holding an item then its will drop it if (assuming an open position):

$$P_{drop} = \left(\frac{\lambda}{\gamma_2 + \lambda}\right)^2 \tag{2}$$

  - if λ is much smaller than γ_2 the likelihood of dropping an item is low.
    - area is not dense enough with items so doesn't want to drop the item.

Basic Ant Colony Clustering Model



Basic Ant Colony Clustering Model

The model can be easily extended to cater for multiple object types

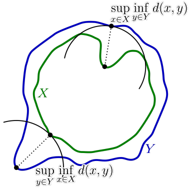
- Simple record a distinct λ_j for each item type
- Then equations (1) in (2) can be used based on the item type encountered to held

Generalized Ant Colony Clustering Model: Lumer–Faieta Algorithm

Lumer and Faieta generalized the basic model of Deneubourg *et al.* to cluster data vectors with real-valued elements for exploratory data analysis applications.

- As with any clustering algorithm we need a metric d to determine the distance between elements. Since we are working with elements in $\mathbb{R}^n$, a simple euclidean distance norm is sufficient
  - What if elements we wanted to cluster where sets? We could use the Hausdorff distance
    - let X and Y be two non-empty sets in $\mathbb{R}^n$

$$d(X, Y) = max\{sup_{x\in X} inf_{y\in Y} d(x, y), sup_{y\in Y} inf_{x\in X} d(x, y)\} \quad (3)$$


$\sup_{x\in X} \inf_{y\in Y} d(x,y)$
X
Y
$\sup_{y\in Y} \inf_{x\in X} d(x,y)$

Generalized Ant Colony Clustering Model: Lumer–Faieta Algorithm

Once you have decided on a distance metric, we have two aims

- Intra-cluster distances are minimized; that is, the distances between data vectors within a cluster should be small to form a compact, condensed cluster.
- Inter-cluster distances are maximized; that is, the different clusters should be well separated.

Generalized Ant Colony Clustering Model: Lumer–Faieta Algorithm

The clusting approach of Lumer and Faieta is as follows:

- Assume we have k data points.
- Randomly distribute the data points over a $N \times N$ grid
- α Ants are then randomly places on the grid.
- Each ant moves independently (and randomly), and makes a decision about where to pickup or drop an item based on the following probabilities, where the item the ant is considering dropping or picking up is denoted as $\mathbf{y}_a$:

$$P_{pickup}(\mathbf{y}_a) = \left(\frac{\gamma_1}{\gamma_1 + \lambda(\mathbf{y}_a)} \right)^2 \tag{4}$$

$$P_{drop}(\mathbf{y}_a) = \begin{cases} 2\lambda(\mathbf{y}_a) & \text{if } \lambda(\mathbf{y}_a) < \gamma_2 \\ 1 & \text{if } \lambda(\mathbf{y}_a) \geq \gamma_2 \end{cases} \tag{5}$$

Generalized Ant Colony Clustering Model: Lumer–Faieta Algorithm

The way you determine the the local density, of the data point $\mathbf{y}_a$ has a large impact on performance. Lumer–Faieta used a patch size of $p \times p$, set $\mathbf{P}$, centered around the current ant position to guide the density calculation as follows:

$$\lambda(\mathbf{y}_a) = max\{0, \frac{1}{p^2} \sum_{\mathbf{y}_b \in \mathbf{P}} \left[1 - \frac{d(\mathbf{y}_a, \mathbf{y}_b)}{\gamma} \right] \} \tag{6}$$

- $\gamma > 0$ defines the scale of dissimilarity between data point $\mathbf{y}_a$ and $\mathbf{y}_b$.
- The constant γ determines when two items should, or should not be located next to each other.
- If γ is too large, it results in the fusion of individual clusters, clustering items together that do not belong together.
- If γ is too small, many small clusters are formed. Items that belong together are not clustered together.

Generalized Ant Colony Clustering Model: Lumer–Faieta Algorithm

As will all CI techniques, there exist numerous variants. Some of which includes

- Different Moving Speeds:
  - The speed is not modelled as movement rate across the grid but rather the accuracy of an ants density measurement
  - Fast-moving ants form coarser clusters by being less selective in their estimation of the average similarity of a data vector to its Neighbors.
  - Slower agents are more accurate in refining the cluster boundaries.

$$\lambda(\mathbf{y}_a) = max\{0, \frac{1}{p^2} \sum_{\mathbf{y}_b \in \mathbf{P}} \left[1 - \frac{d(\mathbf{y}_a, \mathbf{y}_b)}{\gamma + \gamma \frac{v-1}{V_{max}}} \right] \} \tag{7}$$

where $v \in U(1, v_{max})$, and v_{max} is the maximum moving speed.

Generalized Ant Colony Clustering Model: Lumer–Faieta Algorithm

As with all CI techniques, there exist numerous variants. Some of which includes

- Short-Term Memory:
  - Allows the ant to remember a limited number of previously carried items and the position where these items have been dropped
  - If the ant picks up another item, the position of the best matching memorized data item biases the direction of the agent's random walk.
- Distance/Dissimilarity Measures:
  - Utilize difference metrics.

Generalized Ant Colony Clustering Model: Lumer–Faieta Algorithm

As with all CI techniques, there exist numerous variants. Some of which includes

- Pick-up and Dropping Probabilities:
  - Various alteration to the probability of picking up or dropping an item have been proposed
- Heterogeneous Ants:
  - Instead of having all ants having the same control parameters, the parameters are randomly assigned.
  - Or the control parameters change dynamically over time

Division of Labor

Within many insect colonies, a number of tasks are done:

- reproduction,
- caring for the young,
- foraging,
- cemetery organization,
- waste disposal,
- defense.

Task allocation and coordination occurs mostly without any central control, especially for large colonies

- Instead, individuals respond to simple local cues, for example the pattern of interactions with other individuals, or chemical signals

Division of Labor: Using Response Thresholds

Théraulaz *et al.* developed a simple task allocation model based on the notion of response thresholds.

- Response thresholds refer to the likelihood of reacting to task-associated stimuli.
- Individuals with a low threshold perform a task at a lower level of stimulus than individuals with high thresholds.
- Individuals become engaged in a specific task when the level of task-associated stimuli exceeds their thresholds.
- If a task is not performed by individuals, the intensity of the corresponding stimulus increases. On the other hand, intensity decreases as more ants perform the task.

Division of Labor: Using Response Thresholds
Task Allocation

We can utilize similar mathematical models to what was used for clustering.

- Let s_j be the intensity of task-j-associated stimuli.
- Let $\theta_{k,j}$ be the response threshold of individual, k, for performing task j
  - This is not a hard threshold but rather a likelihood indicator of k performing task j.
- The probability of an individual i performing task j is calculated with

$$P_{\theta_{k,j}}(s_j) = \frac{s_j^w}{s_j^w + \theta_{k,j}^w} \tag{8}$$

  where $w > 1$ determines the steepness of the threshold. Usually, $w = 2$.
- The smaller s_j is the less likely task j is to be performed.
- The larger s_j is the more likely task j is to be performed.

Division of Labor: Using Response Thresholds Task Allocation

Stimulus intensity changes over time due to increase in demand and task performance. The amount of stimulus associated with each task is controlled by

- The number of ants performing task j, $n_{act,j}$
  - The cumulative effective of which is indicated by $\gamma_j n_{act,j}$.
- The increase in demand for a task to be performed
  - Indicated by σ_j.

Then the stimuli of task j is updated with

$$s_j(t+1) = s_j(t) + \sigma_j - \gamma_j n_{act,j} \tag{9}$$

- The relative weighting between σ_j and γ_j is vital for long term use

Division of Labor: Using Response Thresholds
Adaptive Task Allocation and Specialization

The fixed response threshold model has a number of limitations

- It cannot account for temporal polytheism, since it assumes that individuals are differentiated and are given pre-assigned roles.
  - Perhaps the threshold of an ant should change at some point in time to simulate temporal polytheism
- It cannot account for task specialization within castes
  - It may be beneficial for a subset to of ants to change their threshold to better fill a niche requirement
- It is only valid over small time scales where thresholds can be considered constant.
  - Consider the division of labor being used by robots, the environment in which they operate can be changed be something as simple as weather.

Division of Labor: Using Response Thresholds Adaptive Task Allocation and Specialization

The simplest approach would be to use a learning coefficient η and a forgetting coefficient θ

- if ant *k* performs task *j* in the next time unit, then

$$\theta_{k,j}(t+1) = \theta_{k,j}(t) - \eta \tag{10}$$

  The ant becomes more likely to perform a task (reinforcing the behavior)

- If ant *k* does not perform task *j*, then

$$\theta_{k,j}(t+1) = \theta_{k,j}(t) + \theta \tag{11}$$

  the ant becomes more less likely to preform a task it ignored in the past (reinforcing the behavior)

Model requires bounding $\theta_{k,j} \in [\theta_{min}, \theta_{max}]$

Division of Labor: Using Response Thresholds
Adaptive Task Allocation and Specialization

The more effective approach is scale to both learning and forgetting depending on how long an ant performs a task.

- Let $t_{k,j}$ represent fraction of time that ant k spent on task j
- Then use the following equation to update thresholds:

$$\theta_{k,j}(t+1) = \theta_{k,j}(t) - t_{k,j}\eta + (1 - t_{k,j})\theta \tag{12}$$

Ant colony Optimization: Applied to Continuous Optimization

ACO can be used if we translate the continuous optimization problem to a binary representation (or directly to a graph).

- However there are the issues of dimensionality, loss of precision, and hamming cliffs can be quite substantially when using floating point binary representation. (Grey coding can take us only so far)

However the ideas of ACO can be used directly for continuous optimization, which is what we will discuss in the continuous ant colony optimization algorithm (CACO)

CACO

The CACO algorithm performs a bi-level search,

- A local search component to exploit good regions of the search space,
- A global search component to explore bad regions.
  - by “bad regions” think regions in which promising positions have not yet been located in.

The algorithm uses n_k ants

- n_l ants are used for local searching
- n_g “ants” are used for global searching

CACO

- Begin by randomly initializing n_r points within the search space.
- Each point $\mathbf{x}_i$ represents a region in the search space.
- Each region fitness is the fitness of their representative point.
- Each region is allocated $\tau_i(0) = 1$ pheromone
- Sort regions in order of fitness.
- From the n_g worse regions
  - 90% are sent for crossover and mutation to find new regions to explore
    - Each region creates 1 offspring.
    - Select a random "bad region", $\mathbf{x}_b$ and for each component j if $U(0,1) < crossprob$ then $\hat{x}_{i,j} = x_{b,j}$ else $\hat{x}_{i,j} = x_{i,j}$
    - Apply Gaussian mutation to the offspring.

$$\hat{x}_{i,j} = \hat{x}_{i,j} + N(0, \sigma^2) \tag{13}$$

where σ is reduced as the time step increases

$$\sigma = \sigma_{max}(1 + r^{(1-\frac{t}{T})^{\gamma_1}}) \tag{14}$$

where $r \sim U(0,1)$, T the total number of iteration. γ_1 controls the amount of non-linearity.

CACO

-
  - 10% are sent for "trail diffusion"
    - Two weak regions are randomly selected to be parents. $\mathbf{x}_i$, $\mathbf{x}_k$
    - and create the following child

$$\mathbf{x}_{child} = \gamma_2 \otimes \mathbf{x}_i + (1 - \gamma_2) \otimes \mathbf{x}_k \tag{15}$$

where $\gamma_2 \sim U(0,1)^d$

- The n_l local ants preform a variant of local search. Each ant, k, selects a region $\mathbf{x}_i$ based on the same probability method we previously used to selected paths, namely

$$p_i^k(t) = \frac{\tau_i^{\alpha}(t)\eta_i^{\beta}(t)}{\sum \tau_j^{\alpha}(t)\eta_j^{\beta}(t)} \tag{16}$$

where the heuristic information is simply the function evolution in the case of maximization.

CACO

- A new point is generated from the selected region by either generating a new point sampled around the region
  - For example a normal distribution
- or using a locally exploitative sub-algorithm
- It would be worth weighting the heuristic component to avoid reusing failed regions.
  - How?