Ant Algorithms

1. Introduction

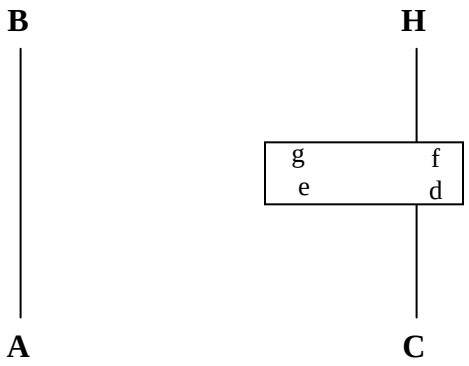
Imagine that you are an ant. You are practically blind, which is a bit of a problem when you are trying to find your way from your home to a source of food or trying to find your way from some food back to your home.
But, we know that ants find their way to and from food all the time. So how do they do it?

It was these type of observations and questions that inspired a new type of algorithm called ***ant algorithms*** (or ant systems).
These algorithms are very new (Dorigo, 1996) and is still very much a research area. It might be the case that they turn out not to produce the results that the initial research suggests they are capable of, but the initial results are promising.

The ant system is a population based approach. In this respect it is similar to genetic algorithms although there is not a population of solutions being maintained. Rather, there is a population of ants, with each ant finding a solution and then communicating with the other ants in the hope it will help them find even better solutions.

2. So how do ants operate?

Consider this diagram.



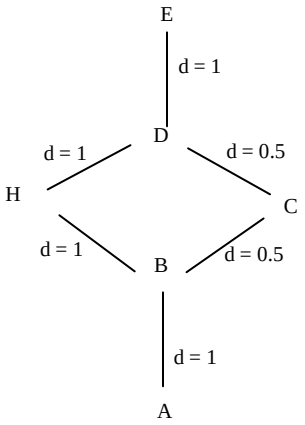
If you are an ant trying to get from A to B then there is no problem. You simply head in a straight line and away you go. And all your friends do likewise.
But, now consider if you want to get from C to H. You head out in a straight line but you hit an obstacle. The decision you have to make is, do you turn right or left?
The first ant to arrive at the obstacle has a fifty, fifty chance of which way it will turn. That is whether it will go C,d,f,H or C, e, g, H.
Also assume that ants are travelling in the other direction (H to C). When they reach the obstacle they will have the same decision to make. Again, the first ant to arrive will have a fifty, fifty chance or turning right or left.

But, the important fact about ants is that as they move they leave a trail of pheromone and ants that come along later have more chance of taking a trail that has a higher amount of pheromone on it.
So, by the time the second, and subsequent, ants arrive the ants that took the shorter trail will have laid their pheromone whilst the ants taking the longer route will still be in the process of laying their trails.
Over a period of time the shorter routes will get higher and higher amounts of pheromone on them so that more and more ants will take those routes.
If we follow this through to its logical conclusions, eventually all the ants will follow the shorter route.

© Graham Kendall - 1/10/2010 - Page 1 of 6

3. A bit more formal

We have described above how ants act in the real world. We can formalise it a little as follows



The above diagram represents the map that the ants have to traverse.
At each time unit, *t*, each ant moves a distance, *d*, of 1.
All ants are assumed to move at the same time. At the end of each time step the ants lay down a pheromone trail of intensity 1 on the edge (route) they have just travelled along.
At *t*=0 there is no pheromone on any edges of the graph (we can represent this map as a graph).
Assume that sixteen ants are moving from E to A and another sixteen ants are moving from A to E.
At *t*=1 there will be sixteen ants at B and sixteen ants at D. At this point they have a 0.5 probability as to which way they will turn. We assume that half go one way and half go the other way.
At *t=2* there will be eight ants at D (who have travelled from B, via C) and eight ants at B (who have traveled from D, via C). There will be sixteen ants at H (eight from D and eight from B). The intensities on the edges will be as follows.

ED = 16, AB = 16, BH = 8, HD = 8, BC = 16 and CD = 16

If we now introduce another 32 ants into the system (16 from each direction) more ants are likely to follow the BCD rather than BHD as the pheromone trail is more intense on the BCD route.

But, as we are interested in exploring the search space, rather than simply plotting a route (in this case we get a form of randomised greedy search), we need to allow the ants to explore paths and follow the best paths with some ***probability*** in proportion to the intensity of the pheromone trail. That is, we do not want them simply to follow the route with the highest amount of pheromone on it, else our search will quickly settle on a sub-optimal (and probably very sub-optimal) solution.

Therefore, the probability of an ant following a certain route is a function, not only of the pheromone intensity but also a function of what the ant can see (which Dorigo calls ***visibility***). Therefore, if an ant is at D (in the above graph) and the pheromone trail is more intense on the route to H then the ant would follow that route if we only considered the pheromone intensity. However, by looking at the distances to the possible routes it would see that the shorter route is to C. It would take this into account when deciding (proabalisticically) which route to choose.
This idea is explored more fully in the discussions on the Traveling Salesman Problem (below).

We also need to stop the pheromone trail building up without bound, because that will ultimately lead to all the ants following the same route. Therefore, we need implement some form of "evaporation" that reduces the pheromone intensity at each time unit. This idea is also discussed below.

4. Ant Algorithms and the TSP

This section gives a brief overview of the seminal paper by Marco Dorigo (Dorigo, 1996).

Ant algorithms are based on the real world phenomena that ants, despite being almost blind, are able to find their way to a food source and back to their nest, using the shortest route. In (Dorigo, 1996) this phenomena is discussed by initially considering ants that only have to walk in a straight line. However, if an obstacle is placed in the way then the ants have to decide which route to take around the obstacle. Initially, there is a fifty-fifty probability as to which way they will turn when they come across the obstacle. If we assume that one route around the obstacle is shorter than the alternative route then the ants taking the shorter route will arrive at a point on the other side of the obstacle before the ants which take the longer route.
If we now consider other ants coming in the opposite direction, when they come across the same obstacle they are also faced with the same fifty-fifty decision. However, as ants walk they deposit a pheromone trail. The ants that have already taken the shorter route will have laid a trail on this route so ants arriving at the obstacle from the other direction are more likely to follow that route as it has a deposit of pheromone. Over a period of time, the shortest route will have high levels of pheromone so that all ants are more likely to follow this route. (Dorigo, 1996) states that this collective behaviour of ants forms an autocatalytic behaviour. That is, there is positive feedback which reinforces that behaviour so that the more ants that follow a particular route, the more desirable it becomes.

To convert this idea to a search mechanism for the Travelling Salesman Problem (TSP) there are a number of factors to consider. These are summarised below and (Dorigo, 1996) has a more detailed discussion.
At the start of the algorithm one ant is placed in each city. Variations on this can be used but, for this discussion, we will assume that the number of ants in the system is equal to the number of cities and that each ant starts in a separate city.
Time, t, is discrete. $t(0)$ marks the start of the algorithm. At $t+1$ every ant will have moved to a new city and the parameters controlling the algorithm will have been updated (n is the number of ants and cities).
Assuming that the TSP is being represented as a fully connected graph, each edge has an *intensity of trail* on it. This represents the pheromone trail laid by the ants. Let $T_{i,j}(t)$ represent the intensity of trail edge (i,j) at time t.
When an ant decides which town to move to next, it does so with a probability that is based on the distance to that city and the amount of trail intensity on the connecting edge. The distance to the next town, is known as the *visibility*, n_{ij}, and is defined as $1/d_{ij}$, where, d, is the distance between cities i and j.
At each time unit *evaporation* takes place. This (which also models the real world) is to stop the intensity trails building up unbounded. The amount of evaporation, p, is a value between 0 and 1.
In order to stop ants visiting the same city in the same tour a data structure, Tabu, is maintained. This simply stops ants visiting cities they have previously visited. $Tabu_k$ is defined as the list for the k^{th} ant and it holds the cities that have already been visited.
After each ant tour the trail intensity on each edge is updated using the following formula

$$T_{ij}(t+n) = p \,.\, T_{ij}(t) + \Delta T_{ij} \qquad (2)$$

where

$$\Delta T_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if the kth ant uses edge}(i,j)\text{ in its tour} \\ & (\text{between time } t \text{ and } t+n) \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

This represents the trail substance laid on edge (i, j) by the k^{th} ant between time t and $t + n$.
Q is a constant and L_k is the tour length of the k^{th} ant.

Finally, we define the transition probability that the k[th] ant will move from city i to city j.

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[T_{ij}(t)]^{\alpha} \cdot [n_{ij}]^{\beta}}{\sum k \in allowed_k \quad [T_{ik}(t)]^{\alpha} \cdot [n_{ik}]^{\beta}} & if\ j \in allowed_k \\ 0 & otherwise \end{cases} \qquad (4)$$

where α and β are control parameters that control the relative importance of trail versus visibility.

For the interested student (and somebody who is willing to do some development) there is a spreadsheet on the web site that implements some of the above formula. The spreadsheet was developed by myself simply as means of being able to cross check values whilst I developed an ant algorithm in a suitable programming language. On a brighter note, at least I know the formulae are correct, or if they are not, then my program is wrong as well!

5. The Algorithm

Below is an informal description of an ant algorithm. This is followed by a pseudo code description.

- At time zero an initialisation phase takes place during which ants are positioned on different towns and initial values $\tau_{ij}(0)$ for trail intensity are set on edges.
- The first element of each ant's tabu list is set to be equal to its starting town.
- Thereafter every ant moves from town *i* to town *j* choosing the town to move to with a probability that is a function (shown in the pseudo code) of two desirability measures.
  - The first, the trail $\tau_{ij}(t)$, gives information about how many ants in the past have chosen that same edge (*i*,*j*)
  - The second, the visibility η_{ij}, says that the closer a town the more desirable it is. Setting $\alpha = 0$, the trail level is no longer considered, and a stochastic greedy algorithm with multiple starting points is obtained.
- After *n* iterations all ants have completed a tour, and their tabu lists will be full; at this point for each ant *k* the value of L*k* is computed and the values $\Delta\tau_{ij}$ k are updated according to formula shown in the pseudo code.
- The shortest path found by the ants (i.e., min L*k*, *k* = 1, ...,*m*) is saved and all the tabu lists are emptied.
- This process is iterated until the tour counter reaches the maximum (user-defined) number of cycles NCMAX, or all ants make the same tour. We call this last case *stagnation behavior* because it denotes a situation in which the algorithm stops searching for alternative solutions.

Pseudo Code

More formally the *ant-cycle* algorithm is:

1. **Initialise**:
   Set t:=0 {t is the time counter}
   Set NC:=0 {NC is the cycles counter}
   For every edge (*i*,*j*) set an initial value $t_{ij}(t)=c$ for trail intensity and $\Delta\tau_{ij} = 0$
   Place the *m* ants on the *n* nodes

2. Set s:=1 {s is the tabu list index}
   For *k*:=1 to *m* do
   Place the starting town of the *k*[th] ant in **tabu**$_k$(s)

3. Repeat until tabu list is full {this step will be repeated (n-1) times}
   Set *s*:=*s*+1
   For *k*:=1 to *m* do

Choose the town j to move to, with probability $p^k_{ij}(t)$ given by equation
{at time t the k-th ant is on town i=tabuk(s-1)}

$$p^k_{ij}(t) = \begin{cases} \dfrac{[T_{ij}(t)]^{\alpha} \cdot [n_{ij}]^{\beta}}{\sum k \in allowed_k \quad [T_{ik}(t)]^{\alpha} \cdot [n_{ik}]^{\beta}} & if\ j \in allowed_k \\ 0 & otherwise \end{cases}$$

Move the k^{th} ant to the town j
Insert town $_j$ in $tabu_k(s)$

4. For k:=1 to m do
   Move the k^{th} ant from $\mathbf{tabu}_k(n)$ to $\mathbf{tabu}_k(1)$
   Compute the length L_k of the tour described by the k^{th} ant
   Update the shortest tour found

   For every edge (i,j)
   For k:=1 to m do

$$\Delta T^k_{ij} = \begin{cases} \dfrac{Q}{L_k} & if\ the\ kth\ ant\ uses\ edge(i,j)\ in\ its\ tour \\ & (between\ time\ t\ and\ t+n) \\ 0 & otherwise \end{cases}$$

where:

$T^k_{ij}(t+n) = p \cdot T_{ij}(t) + \Delta T_{ij}$

5. For every edge (i,j) compute $\tau_{ij}(t+n)$ according to equation $\tau_{ij}(t+n)=\rho.\tau_{ij}(t)+\Delta\tau_{ij}$
   Set t:=t+n
   Set NC:=NC+1
   For every edge (i,j) set $\Delta\tau_{ij}$:=0

6. If (NC < NCMAX) and (not stagnation behavior)
   then
      Empty all tabu lists
      Goto step 2
   else
      Print shortest tour
      Stop

6. Other Applications of Ant Systems

From the description of the ant algorithm, an obvious application is the Travelling Salesman Problem (TSP) but ant algorithms have also been applied to the Facility Layout Problem which can be shown to be a Quadratic Assignment Problem (QAP).

7. Ongoing Research

There are hopefully many other applications for these algorithms and, as we said above, it is an active research area. Even the TSP has many more possible research strands. For example, how many ants should we allow in the system? And is it better if they are evenly distributed around the graph nodes at the start or should they all start at the same node? Dorigo has done some work in this area but there is still more that can be done.
At Nottingham we have applied ant algorithms to the nesting problem (Burke, 1999) and we hope this will be a fruitful area of research.

8. Resources

Marco Dorigo, who did the seminal work on ant algorithms, maintains a WWW page devoted to this subject. If you are interested point your browser at http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html. This site contains information about ant algorithms as well as links to the main papers published on the subject.

9. References

1. E.K. Burke and G. Kendall, "*Applying Ant Algorithms and the No Fit Polygon to the Nesting Problem*", Proceeings of 12th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, 6-10 December 1999, Lecture Notes in Artifcial Intelligence (1747), Foo, N. (Ed), pp 453-464
2. Swarm Intelligence : From Natural to Artificial Systems. 1999. Bonabeau, E., Dorigo, M. and Theraulaz, G. Oxford University Press. ISBN 9 780195 131598
3. Dorigo, M., Vittorio, M., Alberto, C. 1996. *Ant System: Optimization by a Colony of Cooperating Agents*. IEEE Transactions on Systems, Man and Cybernetics – Part B : Cybernetics, Vol. 26, No.1, February 1996, pp 29-41

© Graham Kendall - 1/10/2010 - Page 6 of 6