Computational Intelligence

Unit # 7

Interesting Facts About Ants (Source: Engelbrecht)

- Ants appeared on earth some 100 million years ago, and have a current total population estimated at 10^{16} individuals.
- It is further estimated that the total weight of ants is in the same order of magnitude as the total weight of human beings.
- Most of these ants are social insects, living in colonies of 30 to millions of individuals.

Ant Forging Behavior

- Studies of the foraging behavior of several species of real ants revealed an initial random or chaotic activity pattern in the search for food.
- As soon as a food source is located, activity patterns become more organized with more and more ants following the same path to the food source.
- "Auto-magically", soon all ants follow the same, shortest path.
- This emergent behavior is a result of a recruitment mechanism whereby ants that have located a food source influence other ants towards the food source.
- The recruitment mechanism differs for different species, and can either be in the form of direct contact, or indirect "communication."

Pheromone

- Most ant species use indirect form of recruitment, where communication is via pheromone trails.
- When an ant locates a food source, it carries a food item to the nest and lays pheromone along the trail.
- Forager ants decide which path to follow based on the pheromone concentrations on the different paths.
- Paths with a larger pheromone concentration have a higher probability of being selected.
- As more ants follow a specific trail, the desirability of that path is reinforced by more pheromone being deposited by the foragers, which attracts more ants to follow that path.
- The collective behavior that results is a form of *autocatalytic behavior, where positive feedback about a food path causes that path to be followed* by more and more ants.

Working of Ant Colonies

- The colony's efficient behavior emerges from the collective activity of individuals following two very simple rules:
  - Lay pheromone
  - Follow the trails of others

Working of Ant Colonies

- In a simple case, two ants leave the nest at the same time and take different paths to a food source, marking their trail with pheromone.
- The ant that took the shortest path will return first, and this trail will now be marked with twice as much pheromone (from the nest to the food and back) as the path taken by the second ant, which has yet to return.
- Their nest mates will be attracted to the shorter path because of its higher concentration of pheromone.
- As more and more ants take that route, they too lay pheromone, further amplifying the attractiveness of the shorter trail.

Working of Ant Colonies

- The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate.
- A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate.
- Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution.
- If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Ant System: An Example

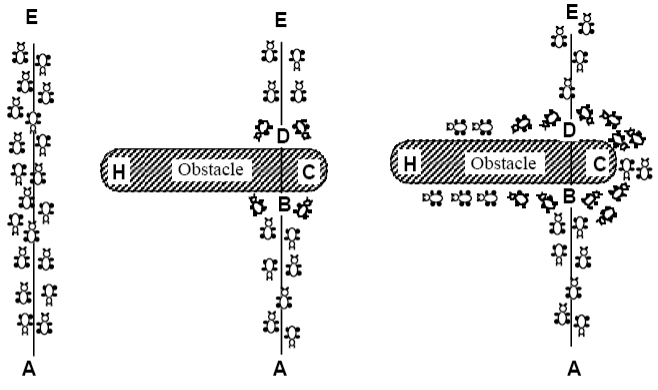


Fig. 1. An example with real ants.

a) Ants follow a path between points A and E.
b) An obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability.
c) On the shorter path more pheromone is laid down.

Ant System: An Example

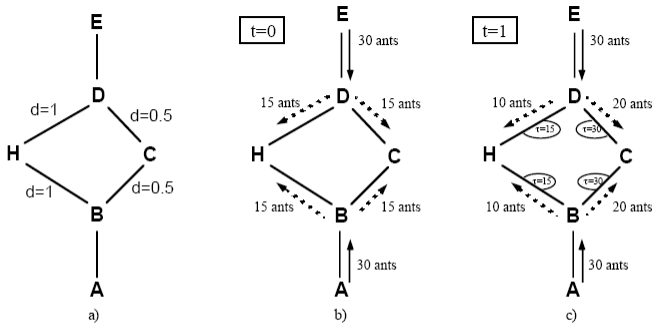


Fig. 2. An example with artificial ants.

a) The initial graph with distances.
b) At time t=0 there is no trail on the graph edges; therefore, ants choose whether to turn right or left with equal probability.
c) At time t=1 trail is stronger on shorter edges, which are therefore, in the average, preferred by ants.

MASON Demonstration

ACO*

- The first algorithmic models of the foraging behavior of ants, as developed by Marco Dorigo in 1992 (PhD thesis)
- A general-purpose heuristic algorithm which can be used to solve different combinatorial optimization problems.
- Not interested in simulation of ant colonies, but in the use of artificial ant colonies as an optimization tool.
- Major differences with a real (natural) ant:
  - Artificial ants will have some memory
  - They will not be completely blind
  - They will live in an environment where time is discrete

* The Ant Systems: Optimization by a colony of cooperating agents by Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni

ACO Algorithm

- Let bi(t) (i=1, ..., n) be the number of ants in town i at time t and let m = sum [bi (t)] be the total number of ants.
- Each ant is a simple agent with the following characteristics:
  - it chooses the town to go to with a probability that is a function of the town distance and of the amount of trail present on the connecting edge;
  - to force the ant to make legal tours, transitions to already visited towns are disallowed until a tour is completed (this is controlled by a tabu list);
  - when it completes a tour, it lays a substance called *trail* on each edge (i,j) visited.

ACO Algorithm

- Let $\tau_{ij}(t)$ be the ***intensity of trail*** on edge (i,j) at time t.
- Each ant at time t chooses the next town, where it will be at time t+1.
- In n iterations, each ant completes a tour.
- At this point the trail intensity is updated according to the following formula

$$\tau_{ij}(t+n)=\rho\cdot\tau_{ij}(t)+\Delta\tau_{ij} \qquad (1)$$

where

ρ is a coefficient such that $(1 - \rho)$ represents the *evaporation* of trail between time t and t+n,

ACO Algorithm

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \qquad (2)$$

where $\Delta\tau_{ij}^{k}$ is the quantity per unit of length of trail substance (pheromone in real ants) laid on edge (i,j) by the k-th ant between time t and t+n; it is given by

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

where Q is a constant and L_k is the tour length of the k-th ant.

- In order to satisfy the constraint that an ant visits all the n different towns, a data structure, called ***tabu list***, is associated with each ant that saves the towns already visited upto time t and forbids the ant to visit them again before n iterations (a tour) have been completed.

ACO Algorithm

- ***Visibility***, η_{ij}, is defined as the inverse of the length of the edge (i,j)
- The transition probability from town i to town j for the k^{th} ant is defined as

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{k \in allowed_k} \left[\tau_{ik}(t)\right]^{\alpha} \cdot \left[\eta_{ik}\right]^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

- α and β are parameters that control the relative importance of trail versus visibility.
- The transition probability is a trade-off between visibility and trail intensity.

Example

	A	B	C	D	E
A	0	5	2	9	3
B	5	0	8	1	9
C	2	8	0	10	2
D	9	1	10	0	4
E	3	9	2	4	0

- Initial Solutions
  - A C D B E 22
  - B A C D E 21
  - E D B C A 15
  - E B C A D 28
  - A D C E B 30
- Given
  - $\rho = 0.6$,
  - $Q = 1$
  - $\alpha = 0.8$
  - $\beta = 0.8$
  - $\tau_{ij} = 0$

Computing τ_{ij}

	A	B	C	D	E
A	0	(1/21)	(1/22+1/21+1/15+1/28)	(1/28+1/30)	0
B		0	(1/15)	(1/22)	(1/22+1/28+1/30)
C			0	(1/22+1/21+1/30)	(1/30)
D				0	(1/21+1/15)
E					0

- Initial Solutions
  - A C D B E 22
  - B A C D E 21
  - E D B C A 15
  - E B C A D 28
  - A D C E B 30

Computing τ_{ij} and η_{ij}

τ_{ij}

	A	B	C	D	E
A	0.000	0.048	0.195	0.069	0.000
B	0.048	0.000	0.067	0.045	0.115
C	0.195	0.067	0.000	0.126	0.033
D	0.069	0.045	0.126	0.000	0.114
E	0.000	0.115	0.033	0.114	0.000

η_{ij}

inverse of the length of the edge (i,j)

	A	B	C	D	E
A	0.00	0.20	0.50	0.11	0.33
B	0.2	0.00	0.13	1.00	0.11
C	0.5	0.13	0.00	0.10	0.50
D	0.11	1.00	0.10	0.00	0.25
E	0.33	0.11	0.50	0.25	0.00

Computing Transition Probabilities

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}]^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

- Numerators only
  - $p_{AB}^{1} = (0.048)^{0.8} (0.20)^{0.8} = 0.024$
  - $p_{AC}^{1} = (0.195)^{0.8} (0.5)^{0.8} = 0.156$
  - $P_{AD}^{1} = (0.069)^{0.8} (0.11)^{0.8} = 0.020$
  - $P_{AE}^{1} = 0.00$
- After Normalization (dividing by the denominator)
  - $p_{AB}^{1} = 0.121$
  - $p_{AC}^{1} = 0.778$
  - $p_{AD}^{1} = 0.101$
- So Ant 1, starting from node A, would decide about the next node based on these probabilities.
- The process is repeated for the other nodes in the sequence as well.
- All the ants follow this process.

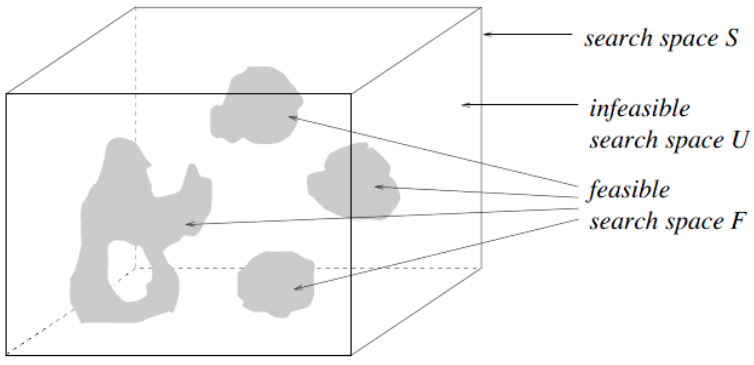
Initial Findings of Dorigo and Di Caro

- SACO works well for very simple graphs, with the shortest path being selected most often;
- for larger graphs, performance deteriorates with the algorithm becoming less stable and more sensitive to parameter choices;
- convergence to the shortest path is good for a small number of ants, while too many ants cause non-convergent behavior;
- evaporation becomes more important for more complex graphs. If $\rho = 0$, i.e. no evaporation, the algorithm does not converge. If pheromone evaporates too much (a large ρ is used), the algorithm often converged to sub-optimal solutions for complex problems;
- for smaller α, the algorithm generally converges to the shortest path. For complex problems, large values of α result in worse convergence behavior.

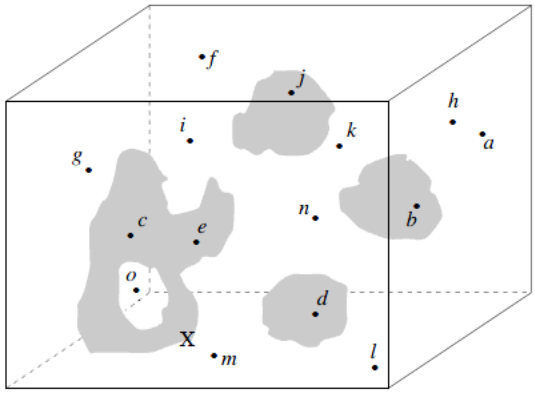
Observations

- From the studies of the simple ACO algorithm, the importance of the exploration–exploitation trade-off becomes evident.
- Care should be taken to employ mechanisms to ensure that ants do not exploit pheromone concentrations such that the algorithm prematurely stagnates on sub-optimal solutions, but that ants are forced to explore alternative paths.
- ACO has an advantage over evolutionary algorithms from similar problems when the graph may change dynamically; the ant colony algorithm can be run continuously and adapt to changes in real time. This is of interest in network routing and urban transportation systems.

Constrained Optimization



Constrained Optimization (Cont'd)



Issues

- The presence of feasible and unfeasible individuals in the population influences other parts of the evolutionary algorithm; for example, should the elitist selection method consider a possibility of preserving the best feasible individual, or just the best individual overall?
- The problem of how to evaluate individuals in the population is also far from trivial.

Constraints Handling

- Several papers have been written on the handling of constraints in Evolutionary Algorithms but the "Do Not Kill Unfeasible Individuals" by Zbigniew Michalewicz is an excellent read.
- The 7 approaches (A-G) discussed in the next slides are taken from Michalewicz paper.

Question # A
Comparison of Feasible Solutions

- How should two feasible individuals be compared, e.g., 'c' and 'j' from Figure (on slide 4)? In other words, how should the evaluation function be designed?

Question A (Cont'd)

- For numerical optimization problems, the evaluation function f for feasible solutions is typically available/given.
- However, for some problems the selection of evaluation function might be far from trivial.
- For example, in building an evolutionary system to control a mobile robot there is a need to evaluate robot's paths. It is unclear, which particular path for a robot should have better evaluation, when we take into account, for example, their total distance, clearance from obstacles, and smoothness.
- For such problems there is a need for some heuristic measures to be incorporated into the evaluation function.

Question # B
Comparison of Unfeasible Solutions

- How should two unfeasible individuals be compared, e.g., 'a' and 'n' in Figure on Slide 4?
- This is a quite hard problem. We can avoid it altogether by rejecting unfeasible solutions (Question # C) or putting a penalty on unfeasible solutions (Question # F).

Question # B (Cont'd)

- Is it possible for an unfeasible solution to have a better fitness than a feasible solution?
- The issue of establishing a relationship between evaluation functions for feasible and unfeasible individuals is one of the most challenging problems to resolve while applying an evolutionary algorithm to a particular problem.

Question # C
Elimination of Unfeasible Solutions

- Should we consider unfeasible individuals harmful and eliminate them from the population?
- This "death penalty" heuristic is a popular option in many evolutionary techniques (e.g., evolution strategies).
- The method of eliminating unfeasible solutions from a population may work reasonably well when the feasible search space is convex and it constitutes a reasonable part of the whole search space.

Question # C (Cont'd)

- Otherwise such an approach has serious limitations.
- For example, for many search problems where the initial population consists of unfeasible individuals only, it might be essential to improve them (as opposed to rejecting them).
- Moreover, quite often the system can reach the optimum solution easier if it is possible to "cross" an unfeasible region.

Question # D
Repairing Unfeasible Solutions

- Should we 'repair' unfeasible solutions by moving them into the closest point of the feasible space (e.g., the repaired version of 'm' might be the optimum 'X' (Figure on Slide 4)
- Repair algorithms enjoy a particular popularity in the evolutionary computation community: for many combinatorial optimization problems (e.g., traveling salesman problem, knapsack problem, etc.) it is relatively easy to 'repair' an unfeasible individual.

Question # E
Replacement of Unfeasible Solutions

- If we repair unfeasible individuals, should we replace an unfeasible individual by its repaired version in the population or rather should we use a repair procedure for evaluation purpose only?

Question # F
Penalization of Unfeasible Solutions

- Since our aim is to find a feasible optimum solution, should we choose to penalize unfeasible individuals?
- This is the most common approach in the genetic algorithms community.
- The major question is, how should such a penalty function Q(p) be designed? The intuition is simple: the penalty should be kept as low as possible, just above the limit below which infeasible solutions are optimal.

Question # F (Cont'd)

- Penalties which are functions of the distance from feasibility are better performers than those which are merely functions of the number of violated constraints.

Question # G
Initialization

- Should we start with initial population of feasible individuals and maintain the feasibility of offspring by using specialized operators?