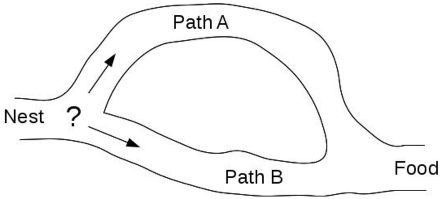
Course: Natural Computing

2* Ant Colony Optimisation

J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

ACO

Biological inspiration: Ant find the shortest path between their nest and a food source using **pheromone trails**.



Ant Colony Optimisation is a population-based search technique for the solution of combinatorial optimisation problem which is inspired by this behaviour.

Marco Dorigo (1992). Optimization, Learning and Natural Algorithms. Ph.D.Thesis, Politecnico di Milano, Italy, in Italian. "The Metaphor of the Ant Colony and its Application to Combinatorial Optimization"

Based on theoretical biology work of Jean-Louis Deneubourg (1987) From individual to collective behavior in social insects. Birkhäuser Verlag, Boston.

What, how?

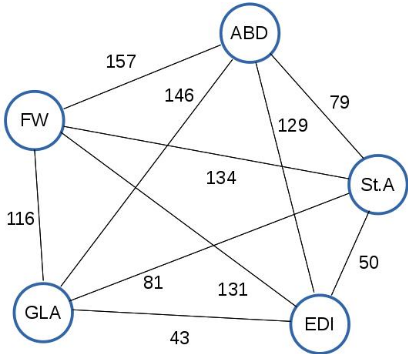
- Real ants find shortest routes between food and nest
- Some species of ants hardly use vision
- They lay pheromone trails, chemicals left on the ground, which act as a signal to other ants: STIGMERGY from stigma (mark, sign) + ergon (work, action), Pierre-Paul Grassé (1959)
- If an ant decides, with some probability, to follow the pheromone trails, it itself lays more pheromone, thus reinforcing the trail.
- The more ants follow the trail, the stronger the pheromone, the more likely ants are to follow it.
- Pheromone strength decays over time (half-time: a few minutes)
- Pheromone builds up on shorter paths faster (it doesn't have so much time to decay), so ants start to follow it.

Artificial Ant Systems

- Pheromones cannot be the only source of information. Their distribution should depend in some sense on the "environment": local heuristics
- Pheromone trail: data structure for "memory" of good solutions
- Ant trails are the solutions
- Pheromone placement by the ant: incorporation of information into the pheromone trail
- Conventions for the algorithm
  - discrete time
  - few foraging ants
  - goal is shortest tour length
  - pheromone laid after the event

So can we apply them to an optimisation problem?

Example: Travelling Salesperson Problem



Find the tour that minimises the distance travelled when visiting all towns.

A solution can be represented as a permutation of the list of cities.

Example: Ant System for the TSP

- Each ant builds its own tour from a starting city
- Each ant chooses a town to go with a probability: This is a function of the
  - town's distance and
  - the amount of pheromone on the connection edge
- The two factors are combined in the Probability Rule (see below) that provides a stochastic decision to go to city j from city i
- Transitions to already visited towns are disallowed till tour completes to guarantee legal tours (keep a "taboo" list)
- When tour completed, pheromone levels are updated on each edge visited

See also: Dorigo and Gambardella (1997) Ant Colony system.
A cooperative approach to the TSP: IEEE TA Evol. Comput. 1:1, 53-66.

Probability Rule

$$\boxed{p(i,j) = \frac{\tau(i,j)^{\alpha}\,\eta(i,j)^{\beta}}{\sum_{k \in \text{Allowed}} \tau(i,k)^{\alpha}\,\eta(i,k)^{\beta}}}$$

- Strength of pheromone $\tau(i,j)$ is favourability of j following i Emphasises "**global** goodness": The **pheromone matrix**
- Visibility $\eta(i,j) = 1/d(i,j)$ is a simple heuristic guiding construction of the tour. In this case it's greedy: The nearest town is the most desirable (seen from a local point of view)
- α and β are constants, e.g. $\alpha = 1$ and $\beta = 2$
- $\sum_{k \in \text{Allowed}}$ normalise over all the towns k that are still permitted to be added to the tour, i.e. that are not on the tour already.
- τ and η introduce a trade-off of global and local factors in construction of the tour.

Pheromone representing promising options

- Pheromone trail evaporates a small amount after each iteration

$$\tau(i,j) = \rho\tau(i,j) + \Delta\tau_{ij}$$

  where $0 < \rho < 1$ is an evaporation constant, describing what fraction of much pheromone is left after one time step. Sometimes the evaporation rate $(1-\rho)$ is used instead.

- The density of pheromone laid on edge (i,j) by the m ants at that time step is

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

- $\Delta\tau_{ij}^{k} = Q/L_k$ if kth ant used edge (i,j) in its tour, else 0. Q is a constant and L_k is the length of k's tour. Pheromone density for k's tour.

Ant systems: Algorithm

- Initialise: Set pheromone strength to a small value
- Transitions chosen to trade off visibility (choose close town with high probability — greedy) and trail intensity (if there's been a lot of traffic the tail must be desirable).
- In one iteration all the ant build up their own individual tours (so an iteration consists of lots of moves/town choices/time steps — until the tour is complete) and pheromone is laid down once all ants have completed their tours.
- Remember: We're aiming for the shortest tour — and expect pheromone to build up on the shortest tour faster than on the other tours

Ant systems: Algorithm

- Position ants on different towns, initialise pheromone intensities on edges
- Set first element of each ant's taboo list to be its starting town.
- Each ant moves from town to town according to the probability $p(i,j)$
- After n moves all ants have a complete tour, their taboo lists are full; so compute L_k and $\Delta\tau_{ij}^k$. Save shortest path found and empty taboo lists. Update pheromone strengths.
- Iterate until tour counter reaches maximum or until stagnation (all ants make the same tour)

Can also have different pheromone-laying procedures, e.g. lay a certain quantity of pheromone Q at each time step, or lay a certain density of pheromone Q/d_{ij} at each time step.

Variants: Ant System (AS)

Pheromone trails: Initialise $\tau_{ij} = \tau_0 \ll 1$

$$\tau_{ij} \leftarrow \rho\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_k} & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

Probability rule for ant k $(1 \leq k \leq m)$

$$p\left(c_{ij} \middle| s_i^k\right) = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{im}\in N\left(s_i^k\right)} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & \text{if } j \in N\left(s_i^k\right) \\ 0 & \text{otherwise} \end{cases}$$

c_{ij}: graph edge, s_i^k partial solution of ant k so far incl. arrival to i, N set of possible continuations of s_i^k (e.g. towards j if $j \in N\left(s_i^k\right)$.

see: Dorigo et al. 1991/92 and www.scholarpedia.org/article/Ant_colony_optimization

Variants: Ant Colony System (ACS)

Pheromone trails: Initialise $\tau_{ij} = \tau_0 \ll 1$

Local pheromone update $\tau_{ij} \leftarrow \rho\tau_{ij} + \sum_{k=1}^{n} \Delta\tau_0$ is now replaced by global update (best ant contribution):

$$\tau_{ij} \leftarrow \rho\tau_{ij} + \Delta\tau_{ij}^{\text{best}}$$

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L_{k^*}} & \text{if best ant } k^* \text{ used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

Pseudorandom proportional rule: Use probability rule with prob. q_0

$$p\left(c_{ij}|s_i^k\right) = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{im}\in N\left(s_i^k\right)} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & \text{if } j \in N\left(s_i^k\right) \\ 0 & \text{otherwise} \end{cases}$$

or make a random (admissible) transition otherwise.

see: Dorigo and Gambardella 1997

Variants: Max-Min Ant System (MMAS)

Best ant adds to the pheromone trails (iteration best or best so far)

Initialise e.g. $\tau_{ij} = \tau_{\max}$

$$\tau_{ij} \leftarrow \rho \tau_{ij} + \Delta \tau_{ij}^{\text{best}}$$

Pheromone production by the best ant only

$$\Delta \tau_{ij}^{\text{best}} = \begin{cases} \frac{1}{L_{k^*}} & \text{if best ant } k^* \text{used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}$$

Minimum and maximum value of the pheromone are explicitly limited by $\tau_{\min}$ and $\tau_{\max}$ (truncation).

Pseudorandom proportional rule

$$p\left(c_{ij} | s_i^k\right) = \begin{cases} \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{c_{im} \in N\left(s_i^k\right)} \tau_{ij}^{\alpha} \eta_{ij}^{\beta}} & \text{if } j \in N\left(s_i^k\right) \\ 0 & \text{otherwise} \end{cases}$$

see: Dorigo and Gambardella 1997

Variants: Max-Min Ant System (MMAS)

Comments

- $\tau_{\min}$ is small and guarantees continuous exploration (may be reduced in order to enforce convergence.
- $\tau_{\max}$ causes all non-visited regions to be equally attractive (pheromone update can be restricted to the links that were actually visited, but you will need fast evaporation in this case)
- Theoretical value of $\tau_{\max} = (L^* (1 - \rho))^{-1}$, if L^* is the shortest tour length. Can be used for "optimistic" initialisation. Smaller values can encourage exploration.
- If $\tau_{\max}$ is not very large it can prevent over-exploration of frequently used regions or may be set to the largest pheromone value that is possible in the given ant system.
- In the face of stagnation the paths can reinitialised by $\tau_{\max}$
- In MMAS as well as ACS, the best ant's contribution can also be combined with from other ants.

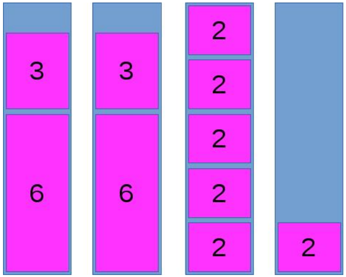
Example: Bin Packing Problem

- Given N items of various weights w_i, and bins of a fixed capacity C
- Lower bound on the number of bins

$$\Lambda_0 = \left\lceil \frac{\sum w_i}{C} \right\rceil$$

- E.g. 10 items of $w_i = 0.6C$, then $\Lambda_0 = 6$, but $\Lambda = 10$ bins are needed.
- Task: Minimise slack $(\Lambda - \Lambda_0)$ or $\Lambda C - \sum w_i$

Greedy: Pack big items first (?)



$(w_i) = (6, 6, 3, 3, 2, 2, 2, 2, 2, 2)$

Bin Packing: Representation

- Bin packing is a grouping problem (TSP was an ordering problem)
- Ants start with all bins empty and instead of walking from city to city, they add item by item
- Items with the same weight are not distinguished
- Pheromone level τ_{ij} can indicate that
    - item i is in bin j (standard) or that
    - items i and j are in the same bin (encourages grouping): Choose new item j by $\sum_i \tau_{ij}$ (and local heuristic) and increase pheromone level each time i and j are in the same bin.
- Taboo rules make sure that the number of items is eventually correct

Bin Packing Algorithm

- Local heuristics: prefer largest item
- Improve solutions by pheromone update based on minimal slack
- As in the $\min - \max$ variant, use minimal pheromone level $\tau_{\min}$ for continued exploration
- Only best ant increases pheromone trail
- alternate global best and iteration-best ant for pheromone update

Bin Packing: Fitness function

- Adding a bin has a drastic effect on the evaluation
- Measure instead how full bins are

$$f(s) = \frac{\sum_{\lambda=1}^{\Lambda} \sum_{i \in \mathrm{Bin}_\lambda} \left(\frac{w_i}{C}\right)^2}{\Lambda}$$

- Minimise number of bins Λ
- Promotes full bins and spare capacity not being spread among many bins.

Bin Packing: Local Search

- For local search, open n_h bins for redistribution of their “free” items
- Items in the remaining bins are replaced by larger “free” items whenever possible.
- Trying all subsets of “free” items is time-consuming. Try instead to replace (e.g.) any two items by any two “free” ones, then two by one “free” item, then one by one “free” item.
- This give fuller bins and smaller items become available which are inserted greedily
- Repeat until no further improvement is possible

Bin Packing: Parameters

- $N = 10$
- $\beta = 2$
- $\rho = 0.75$
- $\tau_{\text{min}} = 0.001$
- $T = 50,000$
- $n_h = 3$

Ant Bin Packing: Conclusion

- Performance is not overwhelming
- Local heuristic is quite cumbersome: ACO seems to direct or initialise local heuristic
- Experience from special case of "triplet" bin packing problem
  - items are either a bit smaller or a bit larger than $\frac{1}{3}C$
  - such that a larger one leaves space for two smaller ones, but two larger ones do not leave space for one small item.
  - deceptive (in terms of local heuristics)
  - AntBin performance is poor in this case!
- ACO an be useful in hybrid algorithms

ACO: Parameters

Main parameters

- Number of ants N (typically $N = 10 \ldots 20$)
- Evaporation constant ρ in pheromone update rule (e.g. $\rho = 0.9$, and even closer for one for longer runs)
- Power α in probability rule (typically $\alpha = 1$)

The following occur in variants of ant systems

- min and max of pheromone level $\tau_{\text{min}}, \tau_{\text{max}}$
- probability of random moves ε

The following are not tunable

- $\Delta\tau_{ij}^{k}$ is fixed by the choice of the cost function L and by the constant Q: Sometimes Q is varied: $Q = 10$ if a new "best" was found and $Q = 1$ otherwise.
- Initial pheromone level τ_0 is not critical because of normalisation, but can be used to implant prior knowledge
- β should be consistent with the choice of the local heuristics

Applications

Intuitively, ACO applications include problem featuring sequential decision marking, such as scheduling, route planing, and any problems that can be brought into this form

- Data stream optimisation
- Cloud computing
- Autonomous driving
- Project scheduling
- Integer programming
- Metro speed profile optimisation
- Bus routes, garbage collection, delivery routes
- Machine scheduling
- Protein folding
- Online network optimisation
- Composition of products

In some of these cases, reinforcement learning tends to be preferable. Many proposed ACO applications were not practically realised.

ACO: (Intermediate) Conclusion

- Cooperative scheme: Building a common solution
- Prone to premature convergence (can be controlled by minimal pheromone level, by adjusting evaporation rate or reinitialising the pheromone levels)
- The strength of ACO is the flexible merging of solutions. This is at the same time a weakness: the components of the solution are chosen independently. In other words: the usefulness of ACO depends on the problem.
- While the original AS is limited to simple dimensions, there are variants of ACO which are quite useful in applications.
- Why do we need several variants? Some elements are important, while other should be used flexibily, to catch the specificity of the problem.

Course: Natural Computing

*2. Brief Tour Through Reinforcement Learning

J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

Overview

- Reinforcement Learning

Reinforcement Learning in a nutshell (20th century)

- Reinforcement Learning aims at enabling an agent to act
- The agent chooses an action based on its current state
- The map from states to actions is called policy
- A state is a vector which is supposed to contain all information that is available to the agent
- The action can lead to a change of the state
  - E.g. the state can be the position of the agent, actions are moves (E, S, W, N), and the agent may need to learn which move to make at which place in order to reach a goal position
- An action was suitable if the agent is given a reward
- The reward may be delayed, i.e. it may arrive only after the agent has performed other actions: What was the reward for?

Metaheuristics and reinforcement learning

- ACO (Dorigo, 1992) is known to be similar to Reinforcement Learning (RL; Sutton, Barto, 1998). An attempt to make this explicit is the Ant-Q algorithm (Gambardella, Dorigo, 1995)
- State transitions
  - The probability rule in ACO determines the probability of a transition from state s to s'
  - The policy in RL states the probability to perform action a in state s that will lead (for deterministic transitions) to state s'
  - The probability rule in ACO is a product of local heuristics and pheromone trail
  - RL uses the sum of immediate reward and the discounted value, but together with the exponential form of Boltzmann exploration this the effect is similar as in ACO.
- For both ACO and RL, many more or less similar variants exist
- Most ACO and some RL variants are Monte Carlo algorithms

Reinforcement Learning and MHO

- In MHO, a state corresponds to a solution, individual, particle; the policy is any method to change the current solutions
- Rewards in RL correspond to fitness in MHO; rewards are often given for part-solutions, fitness for a full solution
- In most RL algorithms a single agent is considered, in MHO usually populations of solutions
- RL requires the problem to be Markovian, MHO is not explicit about this

Reinforcement Learning and MHO

- RL and MHO are similar, but are not the same:
    - RL is an approximation to (local) dynamical optimisation, i.e. the optimisation of a control process
    - MHO is a form of global optimisation, i.e. usually the optimisation of a static problem
- Convergence proofs (in the sense of finding the global optimum) in RL are as impractical as in MHO
- Nevertheless, it is possible to combine RL and MHO in non-trivial ways

Metaheuristics and reinforcement learning

In the past, few attempts were made to combine MHO and RL

- L. M. Gambardella, M. Dorigo, (1995) Ant-Q: A reinforcement learning approach to the traveling salesman problem.
- J. Boyan, A. Moore (2001) Learning evaluation functions to improve optimization by local search.
- A. Nareyek (2003) Choosing search heuristics by non-stationary reinforcement learning.
- A. Eiben, M. Horvath, W. Kowalczyk, M. Schut (2006) Reinforcement learning for online control of evolutionary algorithms.
- R. Battiti, P. Campigotto (2008) Reinforcement learning and reactive search: An adaptive max-SAT solver.

This is not a reading list. You may like to check back if you take an RL course.

Reinforcement Learning (21st century)

- In 21st century policy gradients and deep RL were introduced
- The policy, i.e. the probability what action to take next, is now a parametrised function of the state. Policy gradient methods aim at changing the parameters in order to improve the policy
- The policy as a function of the state can be represented by a deep neural network, i.e. the parameters that represent the policy are weights of a neural network
- Also the map from (e.g.) an input image to the state can be represented by a neural network
- Also the fitness function (reward) can be calculated by a neural network

Implications for MHO?