NAT Tutorial 5: Particle Swarm Optimization

1. Consider **one** of the following problems (or any other one that seems to be interesting) and explain how you would use ant colony optimization to find an acceptable solution: Sequential ordering, classification (e.g. of images), graph colouring, the knapsack problem (or the cutting stock problem), protein folding, the shortest common supersequence problem (for details cf. wikipedia). For this purpose, Dorigo has suggested to answer the following questions:
    a) Define a set of candidate solutions and the set of feasible solutions.
    b) Define a greedy construction heuristic:
        i) What are the solution components?
        ii) How do you measure the objective function contribution of addition a solution components
        iii) Is it always possible to construct feasible solutions?
        iv) How many different solutions can be generated with the constructive heuristic?
    c) Define a local search algorithm:
        i) How can local changes be defined?
        ii) How many solution components are involved in each local search step?
        iii) How do you choose which neighbouring solution to move to?
        iv) Does the local search always maintain feasibility of solutions?

    Answer: Analogous to the AntTSP and AntBin in the lecture. The goal of this exercise is twofold: Get an idea to what type of problems ACO is applicable and recall the main steps of the algorithm.

2. Consider a particle "swarm" consisting of a single member. How would it perform in a trivial task such as the minimization of $f(x)=x^2$? How in a more complex problem? How is diversity produced in a particle swarm of many members?

    Answer: Assume the particle starts with the velocity pointing away from the optimum. Then the personal best would act to retract the particle towards the goal, i.e. the velocity would turn after some time and the particle would head towards the goal. Personal best would loose its influence (the current point is now always "best") and the velocity would slowly decrease (for omega smaller than one). If the velocity reaches zero before the goal is reached the algorithm gets stuck (how would the population help?), otherwise the particle passes the goal, but will be retracted toward the goal as this is now the personal best. Since omega is smaller than 1 it would oscillate around the goal with decreasing amplitude. For this simple problem, it is unlikely that the particle fails unless it did before passing the goal the first time. Iteration-bast is ignored here as it gives no contribution for a single particle.

3. How would you adapt particle swarm optimization to the travelling salesperson problem?

    Answer: Particles are usually searching a continuous space, but TSP is discrete. In Ref. (Kennedy and Eberhart 1997) a discrete version has been proposed. Velocities are still continuous, but positions are discrete and are incremented by a single step when the velocity is above a certain threshold. In this way, the states can directly be used to encode either an integer number (what city to go next) or a binary number (whether one of the (n-1)n/2 links in the graph is used for a tour). It may be useful to

think of a spatial meaning of the states (i.e. such that velocities are meaningful), this is a good strategy for the generalized TSP, where not just cities are to be connected by also rather clusters of cities.

4. Compare the function of the algorithms for particle swarm optimisation and differential evolution.

   Answer: The algorithms are quite similar in structure and areas of application. DE can be considered as a variant of PSO. In particular, the randomisation of the increments or the state vectors is more or less the same, but instead of individual velocities in PSO, differences among two other individuals are used in DE in order to construct increments. While the velocities can learn some goal directedness (similar to the mutabilities in ES), the differences in DE are less likely to represent some goal-orientation, but they can produce exploration in a direct way, whereas in PSO (some form of) the constriction rule is necessary in order to achieve a good balance between exploration and exploitation. Both algorithms can get trapped in a local minimum: for DE “unreasonable” differences (i.e. across random components of the vectors) can be used. For both algorithm this degeneration is not so much induced by the fitness function but by the parameters, in both algorithms an instability of the population can be triggered by a relative increase of the effects from one generation to the next. One does not need to enforce or to wait for convergence, because the best-so-far individual is recorded anyway.

5. Run some example simulations (Start MASON below “Applets and Screenshots”) at http://cs.gmu.edu/~eclab/projects/mason/ E.g.: Ant Foraging, Flockers, HeatBugs and in particular Particle Swarm Optimization (click “Model” to change parameters) Another visualizer is at www.projectcomputing.com/resources/psovis/index.html (PSO only)

6. How are social behaviours in living organisms helpful in developing optimization techniques? Think of examples other than foraging ants, see e.g. www.red3d.com/cwr/ibm.html (meant as an inspiration not as a reading list)

   Answer: All living organisms have solved optimisation problems, although often the optimisation problem is not fixed (as we wish to have it for standard applications) but determined by the environment, i.e. the presence of other individuals/species determines the fitness function. Most interestingly there are many non-trivial applications where the assumption of a fixed fitness function is not appropriate, too, e.g. in optimizing business processes, strategies in games, programming of agents and robots that interact with other individuals (e.g. multi-agent systems, robot soccer).
   Nevertheless, some optimisation problems can be separated from the complex environment of the animal such direct analogies for applications can be made, e.g. pheromones trails as a means to optimise path lengths. Other examples included foraging (bee colony algorithms where different types of individuals produce the solution in a social effort: “explorers”, “messenger” and “gatherers” represent different aspects of the exploration/exploitation problem which can be individully controlled or varied over the course of the run of the algorithm.
   Other examples: Frogs jump “away” when any dark shadow shows. What means “away”? Using the degree of unexploredness as a definition of “away”, the frog analogy can improve exploration in search problems.
   Fireflies: All fireflies are unisex, so that one firefly will be attracted to all other fireflies. Attractiveness is proportional to their brightness, and for any two fireflies, the less brighter one will attract (and thus move) to the brighter one; however, the

brightness can decrease as their distance increases. If there are no fireflies brighter than a given firefly, it will move randomly. Fireflies can also synchronise (i.e. a group is periodically flashing with same same phase and frequency; in this way they can signalise that particular parts of a solution fit well together. This can be used e.g. as a model of language perception (it's then not fireflies but neurons ...)
What about navigation of migrating birds (swarms for robot navigation)? Animals defending their territory (for clustering)? Evolution of parasite-host or predator-prey relations for efficient strategies?
Other related examples (that not really from living organisms) are: Harmony search, formation of river networks, simulated annealing.
Sometimes it is not the analogy which is used in natural computing but the mechanism itself, e.g. the recombination of DNA can be used directly as a computational mechanism.

7. Considering the application of the particle swarm optimization to the travelling salesperson problem, how would you encode a bin-packing problem (see the ACO lectures) in an particle swarm?

   Answer: The TSP was discussed in lect. 15. Bin-packing is very similar. Like in AntBin (Lect. 10), where pheromones tau_b(j) indicate the preference for an item of size j in bin b. What are the required operations?
   – a “position” is an assignment of items to bins
   – a “velocity” is an change of the position (stuffing in some items here, removing elsewhere some items: This might not be possible and needs a procedure closure:
   if an item is to be removed that is not there: ignore
   if an item is to be added that does not fit in: use e.g. first fit decreasing (lect. 10) which is used anyway in AntBin for local search. In the worst case, add more bins.

   Subtraction (position – position) operator: pairwise compare all bins and take idfferences
   Addition (position + velocity) operator: see above “velocity”
   Addition (velocity + velocity) operator: application of one “velocity” after the other.
   Multiplication (coefficient times velocity) operator: possible is a (for a multiplication by 0.5, e.g.) not to perform the operations (removing and adding) in half of the bins. But this multiplication is often not used: drawback: lack of parameters which can be compensated by GA-like operations (that come with parameters) which introduce diversity and compose building blocks.
   After these operations are define, run PSO as usual, making sure that diversity is well-balanced e.g. by measuring size of swarm.

8. Discuss the combination of PSO and particle filters. See Lectures on PSO and the mentioned paper:
   G. Tong, Z. Fang, X. Xu (2006) A particle swarm optimized particle filter for nonlinear system state estimation, http://portal.acm.org/citation.cfm?id=1389095.1389104

9. Go through the ressources at bingweb.binghamton.edu/~sayama/SwarmChemistry/ in particular read Sayama’s article on “Hyperinteractive Evolutionary Computation”