NAT-DL Self-study questions and answers: Applications and other problems Set 7 (week 9)

1. Two main paradigms in computer science are programming and machine learning. Is natural computing different from both or part of either?

   Answer: Obviously, MHO (even GP) is not part of "programming" in its traditional sense, but should be considered as part of ML, which can be generally described as minimisation of objective functions that are implied by data. While ML is more concerned with the formulation and exploitation of the structure, MHO is more focussed on the (global) optimisation aspect, but as we have seen in the CGA or in the ACO-RL connection, there are clear overlaps. In addition, the boundaries between (machine) learning and (computer) programming are beginning to blur, so MHO, i.e. in particular GP, may have the potential to connect the two by producing programs that are incorporating information extracted by ML methods from data, but this is future work.

2. What are the main differences between MHO and

   a) Machine learning?

   Answer: MHO does not use a data set, but requests data. ML often studies the relation between a sample and the underlying distribution, which is rarely (apart from ACO) the case in MHO. ACO can perhaps be considered as an RL method in the disguise of an MHO algorithm, which does not imply that it works particularly well in problems that are more complex than those studied with the basic RL algorithm.

   b) Reinforcement learning?

   Answer: MHO is quite similar, but usually uses populations, and RL usually single agents, RL is an approximation of Dynamical Optimisation, while MHO is aiming at static optimisation (there is considerable overlap, though)

   c) Artificial neural networks?

   Answer: ANNs are basically ML methods , see above, but consider also the recent hybrids of MHO and ANNs, see below.

   d) Mathematical optimisation?

   Answer: Mathematical methods are often based on assumption that do not hold in practice, at least not in a strict sense. E.g. Linear Programming is very powerful, but requires linear constraints and cost function. If the assumptions are lessened then the methods are usually less powerful. Mathematical methods are typically local, the question is of course whether MHO methods do better for any non-trivial problems than random search (which can be considered as a mathematical method).

   e) Biological evolution?

   Answer: This is the odd-ball in the list of sub-questions. In principle MHO should have much in common with natural evolution, but although there are bits and pieces in some of the algorithms, there is not much we have learned about evolution from using MHO algorithms. Areas that could profit (either way), is the role of neutral mutations, the capability to co-optimise fitness and representation (the envisioned level-3 memetic algorithms are not really existing as of today), epistatics, group evolution (i.e. non-trivial swarm intelligence), etc.

   What combinations of methods from any of these with MHO would seem promising?

   Answer: In many cases the answer could be that the respective method is a local search. Recent combinations of RL or ANNs and MHO (week 8) show that there are

now ways to combine learning and MH search in a way that is more close to the idea of Metaheuristics.

Obviously, the answers above do not do justice to the respective fields, and serve only the goal of contrasting MHO methods to other approaches.

3. In the discussion of PSO, a few static topologies, such as ring, mesh, hypercube, complete graph, were mentioned. An adaptive topology may be seen as a dynamic graph where the set of edges or nodes is updated during the search. What are the advantages of using an adaptive topology? What events can be used to cause any changes in the topology? [From E. Talbi, Metaheuristics]

   Answer: We have not talked much about non-trivial topologies or islands so far. So this is a good opportunity to check what benefit may be derived from these approaches: Basically it is about diversification of the search, but there is no guarantee that all subgroups of the population behave differently, especially if the topology (or network or island) does not fully isolate the subgroups of the population. Differently from just more noise, a topology enables in principle to follow several evolutionary paths in parallel, although it should be considered that convergent evolution is not rare in MHO.

   For different algorithms connection or topological neighbourhood can mean that the two solutions can be crossed-over, one can select the other as global best or contribute to the same pheromone trail. Links can have weights that show the likelihood of being retained or cut, or as a probability of being used.

   Adaptive topologies can be used in several ways:

   a) to change granularity, i.e. how many groups (or islands or loosely connected patches of the graph that that describes the connections within the population),
   b) to change the level of interaction (how many neighbours are considered?),
   c) to change the clustering in the graph (more tree-like or more clique-like, a bit similar to the previous, but network theory tells it's not the same),
   d) the diversity within the population (there might be "scouts" for exploration that are not much connected and "gatherer" (for exploitation) that are strongly interconnected,
   e) anything else.

   Now, the question is whether (e.g.) a particle that gets the information about a new best, should approach this new best position, move away from it, or create or cut (or change the strength of) a link. One way to deal with this, is to run a meta-algorithm that learns what network-edit should be done at what event (or at no event) in the learning process. There are a many options:

   – cut any connections between identical individuals
   – weaken connections between similar particles
   – remove connections that are "redundant" (e.g. the third edge in a triangle) to given more flexibility
   – add links randomly to generate rare exchanges (or likewise transport individuals to other islands)
   – start with unconnected particles, then gradually add connections until all particles are in one "giant" connectivity component or until all particles are neighbours of each other
   – connect high-fitness individual among each other, disconnect low-fitness individuals
   – connect to high-fitness individual, disconnect from low-fitness individuals

Note, that in some of these case we would get directed connections which are often more interesting, but may affect also stability.

4. As a social experiment, you are starting a new wiki similar to Wikipedia, but with an upper limit for the total number of entries. If a new entry is created, then the one with the lowest fitness is removed. You have information about the number of readers of each entry, the number of edits and whether text was added, changed or deleted, whether images were added, and possibly also about other aspects. How would you design a fitness function such that this new Wiki remains interesting both for contributors and readers for a long time? What other choices would you make: What limit would you set for the number of entries? Would you limit edits or views based on user identification? Anything else? For discussion in class.

   Answer: This is an open-ended question. It is a bit similar to natural evolution in the sense that survival functions as the main goal for the contributors as well as at the level programming of such as system. On the other hand, we ask for a fitness function that is to be adopted to achieve this goal, which relates to the interesting question of setting up individual fitness functions for the agents in a multi-agent system, where the global fitness is given, but the breakdown to local fitness functions is still to be achieved. A similar experiment was the 1 million pixel Reddit place (see YouTube). Another motivation behind this experiment is the distribution of a fixed amount of wealth in a society.

5. Consider, for the following domains, applications of metaheuristic algorithms, explain fitness, encoding, and details of the algorithm. Make appropriate assumptions on the target domain. Consider in particular whether a formulation as a MOO is advisable.

   a) Trading strategies for automatic agents at a stock market
   Answer: Fitness: gain, variance (so it's MOO), encoding: a vector of data, important data features should be found by the algorithm, this would give it an edge on other automatic trading agents. Algorithm is otherwise a standard GP, e.g. CGP

   b) Game playing agent for a game like chess
   Answer: Fitness: win/loss (it might be possible to see MOO, e.g. if playing against the agent should be maximally rewarding: it could try to optimise length of game too, or let you win after many moves, but not after few etc.). To capture the complexity of the game, we would need to include a neural network here. It is an option to design only the network with the MH, or one may like to aim at a representation of the strategy space by the neural network, and have then MH agents in strategy space.

   c) Shaping of an antenna that is efficient over a wide range of frequencies
   Answer: Fitness: Transmission distance for different angles and bands, again an MO problem as we don't know which will eventually be important. Here many encodings are possible. E.g. a generation program (GP), of a parametrisation of shapes. Or f the antenna consists of a bent wired, then a vector of bending angles and length between the bends.

   d) A heating-ventilation-air-conditioning system for a large building
   Answer: Fitness: cost, fluctuations, user satisfaction; again MOO. The question is, how to get the dynamic aspect into algorithm. One could try something like ACO or an RL-like approach.

   e) Coordination of a group of robots to lift a large object

   Answer: Fitness: no MOO, we can be happy if the robots make it and don't need too many attempts. My guess would be the positions and movement directions of the robots for each time step for discrete. Or RL-like? Or control programs?

6. Summarise the conditions for practicable applicability of MHO.

   Answer: This is just meant to recap a few key observations

   a) Small- to medium-scale problem (which may be a sub- or super-problem of a larger problem, such as feature selection or hyperparameter tuning)
   b) Compared to its size the problem relatively complex (e.g, NP-complete problems such as scheduling and allocation problems, but also design problems), otherwise empirical gradient methods (Nelder-Mead downhill simplex) may be preferable.
   c) Little reliable information about relevant problem properties beforehand (e.g. dynamic problems, such as finding trading strategies in finance)
   d) Complex goal structures as in multi-objective optimisation for negotiation with stakeholders
   e) Relatively small number of fitness evaluations (or labelled data) that are the main source of learning about the problem
   f) Availability of resources for parameter tuning and hybridisation

7. Perform a quick search for recent practical applications of natural computing.

   Answer: There are as many answers as hits in you favourite search engine. Here are some points that I like to mention in this context:

   a) Several years ago, even washing machines featured “intelligent neuro-fuzzy” controllers, which was meant to assure the customers that a machine as complex and highly developed as the present one, was still easy to handle. There might be a similar way to produce interfaces between users and machines by GP?
   b) Financial data are often driven by new information, trends, or instabilities that affect the market for a short time. Approaches relying on massive data may be an overkill given that only a short time window is relevant for what happens and the no long term solutions will be identifiable.
   c) More and more autonomous vehicles will be out there in near future while for automatic cars probably a centralised (or multi-centred) system will take over the control and coordination, for smaller vehicles and drones (e.g. in delivery and in agriculture) swarm behaviour may be provided a useful guidance for the robots.
   d) Cyberphysical systems, microrobots, and in particular nanorobots have limited computational resources and computational resources tend to use up a lot of energy, so MHO methods could help to face computational challenges in the real world, by some exploration rather than extended computational modeling.

8. Prepare a (very brief) business case for a natural computing application of your choice which is to be discussed in your group. How could you defend your case against any foreseeable counterarguments?

   Answer: This question is not meant to think too much of the general problem of starting a business, but about the technological readiness levels. Usually grouped from 1 to 10, with 1 being a theoretical question (e.g.: Can the movements of bird flocks be described by simple computational rules?) and 10 a finished and adoptable project in the development department in a company (or analogue). MHO seems to be indifferent to this scale, because of its ease in adaptation to many problems. It is nevertheless clear that many theoretical questions (mainly the “alignment” question that we mentioned together with the NFL theorem, and the Building Block Hypothesis) are still open while level 10 may have been reached only in financial applications. You may like to consider the examples from the previous question in order to develop you own thoughts here.