NAT-DL: Self-study: NFL and bio-inspired computing Set 5 (week 7)

1. What are the implications of the NFL theorem for MHO? In order to discuss this question, assume that you are producing a comprehensive MHO software package that is meant to solve a large variety of problems for its users e.g. in logistics, IT, or finance.

   Answer: This is an opportunity to repeat the statement and to discuss the implications, see lecture slides for more information, but the question also refers to the differences between applications areas, i.e. in some areas (like logistics or in some IT applications) there are re-occurring regularities that the algorithm can exploit, e.g. when deciding about a tour length in a typical practical problem, the "visibility" (i.e. the length of the next leg) will contain at least a trace of information about the total tour length. Although information is not fully reliable, it will still give an advantage over random tours. Likewise, there are principles (e.g. removing self-crossings will lead to a better tour in a planar TSP) that can lead to an advantage given that the algorithm is used only for this particular type of problems. On the other hand, in finance, the existence of any such structure, e.g. in stock market data, is a way to make money, so on average algorithm cannot work well. The trick here would be to choose an algorithm only temporarily (e.g. a week or two) and to change it before averaging as well as the reactions of the competitors make is useless again. So the software should provide a large variety of algorithm such that the user can flexibly switch among them.

2. There are a number of algorithms, search spaces and conditions that may seem to provide an escape from the No-Free-Lunch theorem. Discuss whether any of the following cases can lead to a free lunch, and also try to find other potential exceptions. Here. the question is not whether any optimisation problem actually gets solved, but whether there is a chance to get at least a small advantage over a random walk on average over all problems. Consider, however, that, if in some of the cases below the NFL theorem does not apply, then this does not automatically mean that there is a free lunch.

   a. Resampling algorithms, i.e. algorithms that do not check whether they have sampled a point already.

   Answer: Some algorithms will resample less frequently than others, so there is a clear advantage for these algorithms. This is the reason that resampling algorithms are excluded in the NFL theorem, because the avoidance of resampling is a trivial advantage an algorithm can have.

   b. Deterministic algorithms compared to stochastic algorithms.

   Answer: A stochastic algorithm may less easily be protected from resampling, but if we assume that the algorithms are non-resampling there is no advantage for either. It should be considered, however, that in complexity theory the stochastic algorithms are considered as parallel algorithm. For example, the famous NP problems can be solved by a non-deterministic algorithm in polynomial time, because this algorithm is assumed to move to all possible solutions (even to exponentially many) in parallel, so they can search a hypercube in N steps, see Question 4e) below. We are considering here not the abstract idea, but a realisation of a non-deterministic algorithm that follows only one trajectory of the abstract algorithm. As we can describe this trajectory also by a (possibly very complex) deterministic algorithm, there is no difference between the two cases.

   c. The *co-evolutionary* case where two algorithms are searching independently on the same problem and exchange the information about the next state and the fitness of the state after each iteration.

   Answer: This is one exception to the NFL theorem, see the paper by Wolpert and Macready (2005) on coevolutionary free lunches. The paper is not easily readable, but the main idea is quite simple: Say algorithm A is using the information from algorithm B, but

B does not care about *A*. Then, either *A* finds the solution on its own while *B* is still searching. Alternatively, B finds it first in which case A can use this information to immediately move to the solution, such that on average A will perform better than B. This works for all problems. Although this result seems quite trivial, it is interesting as it represents an actual exception to the NFL theorem. Co-evolution occurs also in nature, but there it does not seem to provide a free lunch. The fact that the "arms race" in natural evolution presents all species with an increasing difficulty, is, however, a feature that can be used also in algorithms. For example, algorithms (or even two instances of the same algorithm) can compete in games to gradually improve their fitness (see also the self-play in Alpha-Go). In this way the problem is initially very easy, so that any solution will do quite well, can be improved later. If the algorithm plays against human player it may not win at all initially, such that the progress is questionable. Of course, a smoother fitness function can be obtained also in the latter case by counting fitness based on for how long the evolving player has survived or how many points it has achieved. A fitness function that only provides binary information (such as about win and loss), may not be suitable also in other problems.

d. An unknown algorithm produces samples and the tested algorithm is supposed to predict which sample the unknown algorithm will choose next. Can there be an algorithm that is better than others in emulating the unknown algorithm when averaged over all unknown algorithms?

Answer: We would average here over the unknown algorithms rather than over just problems. For finite state spaces, each problem can also be produced by an algorithm such that there is no exception here, i.e., we would need to specify a distribution over algorithm in order to get a free lunch. For infinite spaces or for theoretical algorithms that include continuous numbers, there may be free lunches, see the next two questions.

e. The case of a continuous search space, such as PSO.

Answer: Over a continuous search spaces the set of problems is uncountably infinite, so a uniform distribution over problems does not exist. The idea of convergence of PSO requires the definition of an approach to the global optimum as close as required. However, the problems can have arbitrary fitness fluctuations also on scales that are smaller than the required precision, such that PSO would not be able to take advantage of the free lunch that may be possible here.

f. The case of an unbounded discrete search space (E.g., the set of natural numbers)

Answer: Here a uniform distribution over problems cannot be defined, so that the NFL theorem is not applicable. There exist papers that show that in these cases a "free lunch" is actually possible.

g. The case of a small discrete problem where all fitness values can be evaluated.

Answer: If all fitnesses are known to all algorithms, then some algorithms (not the good ones) may not report the optimum, even if they have the information about it. Although this seems like an exception to the NFL theorem, we need to implement the sampling procedure as described in the lecture, i.e., the (non-resampling) algorithms have to ask for a different sample in each time step. In this way, they cannot not report that they found the solution, i.e., the NFL theorem is not applicable. We could also say that if all samples are known to the algorithms then clear information about the problem is given. This means, that instead of the distribution over the full problem space, the distribution is now concentrated on a single problem, which also contradicts the assumptions, but know w.r.t. the uniform distribution over problems. If, however, the algorithms are not assumed to sample all states, then we are clearly

within the scope of the NFL theorem.

h. Memetic algorithms: Algorithm A chooses one algorithm from a set of algorithms, e.g. A can choose either A_1 or A_2 after sampling a certain number of fitness values.

Answer: The initial sampling of the fitness values has the cost that later fewer fitnesses are available. Also, there is no guarantee that the later samples confirm the information extracted from the first batch.

i. The case of an algorithm with critical parameter values (see e.g. DE or PSO) as compared to other parameter settings.

Answer: Critical parameter settings imply a good balance of exploration and exploitation, in general this depends on the problem (see point k) below. However, for PSO and DE we seem to be able to determine criticality also without assumption about the problem. The point is that criticality works only if we have an unbounded search space, then the prior over problems expresses the information that the solution is more likely to be nearby than arbitrarily far away. In other words, critical algorithms perform a scale-free search. This means, they assume a distribution over problems where all scales are equally likely. This is a reasonable assumption and presents an exception to the NFL theorem, simply because the NFL theorem is not applicable to cases where no uniform distribution over problems exist.

j. A genetic algorithm with a diverse population compared to the case of a redundant population.

Answer: A redundant population implies resampling, see point a) above.

k. An algorithm with a perfect balance of exploration and exploitation.

Answer: This algorithm would outperform other algorithms. However, the "perfect balance" depends on the problem. Therefore, the algorithm would need to know what type of problem it is applied to, which implies a non-uniform distribution over problems, or it would need to find out by using up a number of fitness evaluations. This is a disadvantage that averages out over all problems. Although no exception from the NFL theorem is found here, it is practically a good idea to maintain this balance as it is often more easily usable for the control of the algorithm than other known or explored features of the problem.

l. The set of all real-world optimisation problems that have ever been studied.

Answer: We are asking for an algorithm that works better than other on average over these problems. However, we do not know how to characterise "real-world" problems in general. Similar to the "arms race" mentioned above, we would assume that problems studied so far are the more easy problems out of the set of all problems, such that the NFL theorem does probably not apply.

3. Try to adapt the island model of GA to ACO algorithms. Given a set of ant colonies connected by a given topology, which strategies can be applied to exchange information between the colonies? Specify the integration procedure of the information received in the destination colony. [From E. Talbi, Metaheuristics]

Answer: In the previous question the types of interactions or topologies were discussed, now we are focussing on integration. The simplest way would be to add the pheromone trails from each of the sub-colonies to the other. A weighted addition can be considered with weights implied by the pheromone update scheme (as this does not have to be the same in the sub-colonies), e.g. pheromone that in one sub-colony was obtained by a single ant may have a higher weight than the pheromone that was accumulated by several ants in parallel. It could also be considered whether ρ is the same (then weigh by the theoretical maximal value of $1/(1-\rho)$),

or whether the trails were normalised.
Instead of addition also other options should be considered: It is possible to multiply (and renormalise) the trails, which would lead to a sharper result. Obviously (remember that ACO is similar to Bayes), it can happen that all paths from a node can get zero probability in this way, but this can be avoided by the τ_{min} rule. One could aslo use a fuzzy multiplication: $(a(b^{\mu}) + (a^{\mu})b)/2^{\mu}$ for μ in [0,1], i.e. μ=0 for addition, μ=1 for multiplication. Other possibilities include max of the two, or a type of crossover (for some part solution take information from population 1, for other ones from population 2), or for more than two populations also a kind of DE rule becomes possible. Again, admissibility of the resulting trails needs to be checked. The preference is to be given to methods that increase diversity, as this is often a problem in ACO (what about μ<0?).

4. A Lévy flight is a random walk with a diverging variance. Lévy flights were shown to provide suitable models for animal foraging behaviour. They have also been used in several MHO algorithms, e.g. Cuckoo search. Under what conditions is this type of exploration useful?

   Answer: This is just a repetition of what was mentioned in an earlier unit, but the idea of a Levi flight is too important, not to discuss it further. Not that actually Levi flights do not exist, they need to be truncated (or exponentially cut-off) in order to be realisable in an algorithm (or realised in nature). This means that large values are not arbitrarily large and small values cannot get an infinitely large probability. Note also the Levi flights appear also in a particle swarm if the particle is subject to critical dynamics.
   Lévy flights are scale-free, i.e. if the scale changes the distribution (apart from any cut-offs) is the same.
   This is also the condition of applicability: If the problem is scale free then Levi flights are useful. Often scales are known, but hey would correspond to cut-offs: discretisation of the problem: lower cut-off; size of search space: upper cut-off. If there is, however, also another scale (e.g. obtained by landscape analysis: correlation length or typical distance between to local minima etc.), then perhaps a Levi flight will not be needed.
   What should we do if there are many different scales in a problem? It depends on the problem: Are the scales different in different parts of the search space (use a hyperheuristic), in different directions (learn a direction), are some scales more relevant to the fitness (use an adaptive algorithm) or are the scale just wildly combined (then stick with the Lévy flight).

5. Choose <u>one</u> of the following papers (or, if you prefer, a similar one, e.g. from this list https://en.wikipedia.org/wiki/Table_of_metaheuristics). There is no need here to read your paper in depth, just make sure that you can briefly explain its main idea in class within 2 mins. You can choose to tell about (or to ignore) the respective biological inspiration or the algorithm. Likewise, you can discuss whether the algorithm is related to any of the main MHO algorithms that we have discussed so far.

   Answer: This task is intended to help to understand the relations among algorithms, not to teach more algorithms. There is no correct solution, so just a few notes are given on how the respective algorithm may be related to the main examples discussed in this course.

   a. Maziar Yazdani and Fariborz Jolai. “Lion optimization algorithm (LOA): A nature-inspired metaheuristic algorithm”. J. Comput. Design and Engin. 3.1 (2016), 24–36.

      First, you need to know that a group of lions (usually several females, their cups and one or more males) is called *pride*. Solution vectors (i.e. the “lions”) in search space are reflected about the centre (the “prey”) of the swarm, similar to a PSO with alpha=4, but not all of them so that they converge to the centre of the swarm from different sides

(lions rounding up their prey). The centre is updated based on the fitness of the approaching "lions", and more toward the direction where the "lion" had a bigger fitness increase, this is how the information gets in the game. As the algorithm would converge quickly also crossover is included as well as an organisation into *prides* (similar to island in a GA). Comparison on benchmark functions is against obscure MHO algorithm not against (e.g.) hill-climbing with restarts.

b. Marko Mitić et al. "Chaotic fruit fly optimization algorithm". Knowledge-Based Systems 89 (2015), 446–458.

   Groups of fruit flies are apparently called *cloud*, *swarm* or even *business*. The algorithm is just a random work about the currently best solution, again like PSO (without personal best and without momentum). The remaining alpha*random in PSO is now not actually random but follows a chaotic trajectory. They try several such maps and compare results on benchmark functions to other likewise chaotic algorithms. The distribution of the points in the chaotic time series is different, so that an algorithm with noise of the same distribution would have essentially the same performance (unless either the chaotic data are worse due to temporal dependencies or the algorithm is applied to the odd problem which it is aligned to and which must exist according to the NFL theorem).

c. Gai-Ge Wang et al. "A new metaheuristic optimisation algorithm motivated by elephant herding behaviour". Int. Journal of Bio-Inspired Computation 8.6 (2016), 394–409.

   We learn that a family group of elephants are called *clan*, which are subdivisions of the *herd*. The update is towards the best in the clan with a random factor and a small alpha (elephants do not jump), whereas the best is moved a bit towards the centre of the clan, but the given equation is not clear. The worst elephant is randomly reset, but also this formula seems flawed (why is there a "1"). Later they add an elitism strategy, but why the best is first moved around and reinstalled by elitism is not clear. Also, the interaction among the clans in the herd is not really discussed.

d. Gai-Ge Wang, Suash Deb, and Leandro Dos Santos Coelho. "Earthworm optimisation algorithm: A bio-inspired metaheuristic algorithm for global optimisation problems". International Journal of Bio-Inspired Computation 12.1 (2018), 1–22.

   This algorithm is a bit similar to the CGA (a theoretical version of GA that we have mentioned in the theory part): solutions are moved to an "opposite" place in the search space, but not (as in (b)) about a centre but with respect to the boundaries, i.e. it the point was at ¾ of the 1D search space [0,1], it is now moved to ¼, but this can also be modified by a random number or done with respect to some dimensions. Pairs of such solutions can be crossed-over or mixed with some fitness-related weighting. The weighting schemes are quite complex, possibly due to the problem that the solutions will develop a tendency to move towards the centre of the search space otherwise.

e. Xin-She Yang, Mehmet Karamanoglu, and Xingshi He. "Flower pollination algorithm: a novel approach for multiobjective optimization". Engin. Optimiz. 46.9 (2014), 1222–1237.

   This is similar to the "cuckoo search" mentioned in the lecture (also by the same author): a simple movement to the current best shifted by a bit of Levy noise: "in many ways, it has some similarity to … cuckoo search". The idea here is that it is now for multi-objective optimisation (in this week's lecture).

f. Seyedali Mirjalili and Andrew Lewis. “The whale optimization algorithm”. Advances in Engineering Software 95 (2016), 51–67.

   Whales “encircle” their prey, so that it seemed to make sense to the authors to do something like the spiral optimisation that we have mentioned under physics-based MH. The “exploitation” is either (with p=0.5) on a shrinking cube surrounding the current best or using the mentioned convergent spiral toward the best (with p=0.5), but I’m not sure why both is needed. Exploration is simply by adding noise. This algorithm seems most close to hill-climbing with occasional reset, which different from my admittedly limited understanding of *pods* (or *schools* or *gams*) of whales.

g. Ekrem Duman, Mitat Uysal, and Ali Fuat Alkaya. “Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem”. Information Sciences 217 (2012), 65–77.

   I’m not really sure about this one in respect of its usefulness for optimisation, also there are no meaningful equations given. Apparently a local search but among subgroups: try to improve a given individual by any direction towards each in a set of other individuals. The subsets are chosen in a cyclic way like in V-formation of migrating ducks.

h. Gai-Ge Wang, Amir H Gandomi, and Amir H Alavi. “An effective krill herd algorithm with migration operator in biogeography-based optimization”. Applied Mathematical Modelling 38.9-10 (2014), 2454–2462.

   This one is quite similar to PSO: There is interaction between particles essentially towards the best, there is movement similar to PSO, and there is diffusion (i.e. randomness is a bit different from PSO). The “biogeography” is simply a local (but random) search.

i. Jose Orellana and Ricardo Contreras. “Bacterial resistance algorithm”. Int. Workconf. Interplay Between Natural and Artificial Computation, Springer (2019), 204-211.

   This one was added in order to show that the proliferation of new algorithms does not cease. I’m not sure why fitness is now “resistance”, but this seems to be due to the metaphor. Apart from the use of a set of three modification operators (although just to ex-change or randomise parts of the solution), there is no novelty here. It is nice that they do say “Results are comparable to those obtained in similar approaches.”, whereas the previous papers presented their results as best within a group of the other algorithms to which it is compared. This is less because of the NFL theorem (since the benchmark functions are the usual standard), but possibly because the algorithm is first tested against variants of its own (which is essentially a parameter optimisation), and the best variant is then tested against other algorithms (rarely standard algorithms), but it is not clear whether the para-meters of these algorithms are also optimised (and how well) with respect to the tasks.

See also: Nader Behdad (2022) Review: Metaheuristic Optimization Algorithms. Int. J. Computer Applications, 184 (30) 33-38.

As well as: Camacho-Villalón, C.L., Dorigo, M. and Stützle, T., 2023. Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: Six misleading optimization techniques inspired by bestial metaphors. *Int. Transact. Operat. Res.* **30**(6), 2945-2971.