Competitive Coevolution in Problem Design & Metaheuristical Parameter Tuning M.Sc. Thesis

Guðmundur Einarsson, Thomas Philip Runarsson & Gunnar Stefánsson

School of Engineering and Natural Sciences, University of Iceland, Iceland.

Abstract

The Marine Research Institute of Iceland has over the last 20 years developed and used the program `gadget` for modeling of the marine ecosystem around Iceland. The estimation of parameters in this program requires constrained optimization in a continuous domain. In this thesis a coevolutionary algorithm approach is developed to tune the optimization parameters in `gadget`. The objective of the coevolutionary algorithm is to find optimization parameters that both make the optimization methods in `gadget` more robust against poorly chosen starting values and tries to reduce the computation time while maintaining convergence. This is important when bootstrapping is performed on the models in `gadget` to reduce computation time. This may also ease the tuning of optimization parameters for new users and may reveal other local optima in the likelihood, which may give hint of model misspecification. The algorithm is tested on functions that have similar characteristics as the log-likelihood functions in gadget and some results shown for the case of modeling haddock.

Heuristics and Metaheuristics

Heuristics in optimization are methods that find approximate solutions to optimization problems. The purpose is to provide good seeding values for exact algorithms and sometime they are the only viable option when the search space is large, multimodality is present in the objective function or the objective function is illconditioned. Other problems also come up where heuristics are used e.g. combinatorial optimization.

Heuristics trade various aspects of exact algorithms for speed, e.g. optimality and precision. Another thing about heuristics is that they are often based on a rule of thumb or examples from nature, which often makes it hard to proof anything for their performance. These methods are sometimes *ad hoc* and don't generalize well to other problems.

Metaheuristics can be summed up in the following definition by Glover (1986)

> A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. The term is also used to refer to a problem specific implementation of a heuristic optimization algorithm according to the guidelines expressed in such a framework.

Note that a metaheuristic can refer to a framework, such as *evolutionary algorithms* and a specific algorithm, such as *differential evolution*.

One of the reasons these algorithms have become popular lately is because they usually scale very well to larger computer architectures, they are easily parallelizable. The reason for this is that many of these algorithms are based on *agents* or *individuals* that cover the search space. These can be independant or have some interactions so the computation for each agent can be calculated on an individual CPU core.

Although metaheuristics may not be directly applicable to deep learning, it has still potential to be used for feature selection and other aspects of machine learning.

Competitive Coevolution

Coevolution is a term from evolutionary biology. It symbolises the change in a biological object triggered by the change in another biological object. It can happen cooperatively and competitively, and it can happen at the microscopic-level in DNA as correlated mutations or macroscopic-level where covarying traits between different species change.

Hillis (1990) is the first person to publish an article on the usage of competitive coevolution as a metaheuristic. His paper has at the time this thesis is written, over 1100 citations. He used the competitive nature of the algorithm to create abnormal/pathological cases to deal with, which helped tune another algorithm for generating minimal sorting networks so it would not get stuck at local optima when dealing with such extremities. This setup has inspired others to do the same, when either tuning optimization algorithms or training/tuning game-playing strategies.

When the species are in some sense symmetrical, i.e. representations of opposing players in some games, then it can occur that no strategy dominates all others, i.e. we have non-transitive players. One of the players can always find a strategy to counter what the other player is doing. This can create cyclic behaviour, and has been studied (Samothrakis et al., 2013). Although our problem is transitive in nature, cycling can still occur (De Jong, 2004) and this further emphasizes the importance of the posterior assessment of the behaviour in the coevolution.

Evolutionary algorithms are special types of heuristics and metaheuristics. They are what is called generic-population based metaheuristics. They are formed from the analogy of evolution and the main aspect borrowed is the survival of the fittest. The methods developed from this framework do not need rich domain knowledge, but it can be incorporated. Some technical terms have special synonyms in the theory of evolutionary algorithms, which are summarized in the following table.

EA term	Regular term
Fitness	Objective/Quality
Fitness landscape	Objective Function
Fitness Assessment	Computation of objective function value
Population	Set of candidate solutions
Individual	Candidate solution
Generation	Population produced during a specific iteration
Mutation	Alter candidate solution independently
Crossover	Alter candidate solution with other solutions
Chromosome	Solutions parameter vector
Gene	Specific parameter in the Chromosome vector
Allele	Specific value of a gene
Phenotype	Classification after fitness assessment

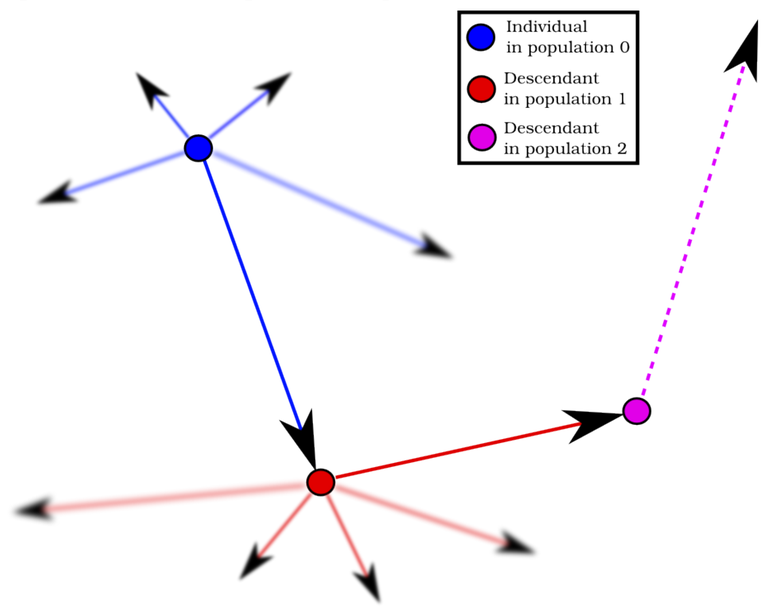
The section Algorithms and Examples summarizes how the algorithm is built using tools from the evolutionary algorithm framework and describes the problems used to test the algorithm.

Algorithm and Examples

The species in the Coevolution consist of *predators* and *prey*. The predators are represented by numerical vectors which are parameters for optimization algorithms and the prey are represented by numerical vectors which represent starting values for the optimization. In general the prey can be any parameter vector which defines a family of function for an optmization algorithm to call.

There is no static objective function, since in each iteration of the algorithm the score for an individual is dependent on the individuals in the competing species. Therefore *relative fitness assessment* (RFA) is used. The RFA is the mean of the score for a certain induvidual against some of the individuals in the competing species and the score is a weighted sum of the number of iterations for each optimization algorithm and the final value from the optimization. This score/fitness then represents how well an individual performs relative to the other individuals in the species.

The fitness is calculated after a generation of new individuals with genetic operators. The best individuals are kept for the following generations, they must be significantly better than the individual they were generated from such that they are chosen. The plot below vaguely describes how the genetic operators work in a plane for a given individual.



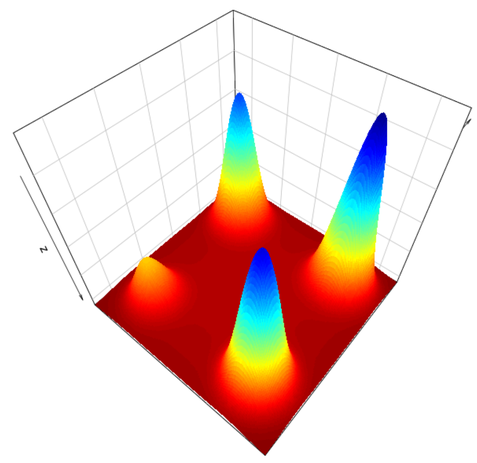
The new potential candidates for the next generation are generated with the genetic operators from differential evolution (Storn et al., 1995). It is a combination of mutation and crossover operators.

It is very tedious to perform the RFA such that all the individuals interact. Therfore a sampling strategy was developed to decrease the number of fitness evaluations. In the thesis it is shown empirically that the

Examples, the examples used were made to portray the algorithm's capabilities to find hard starting values for the predators. One of the examples is the following function.

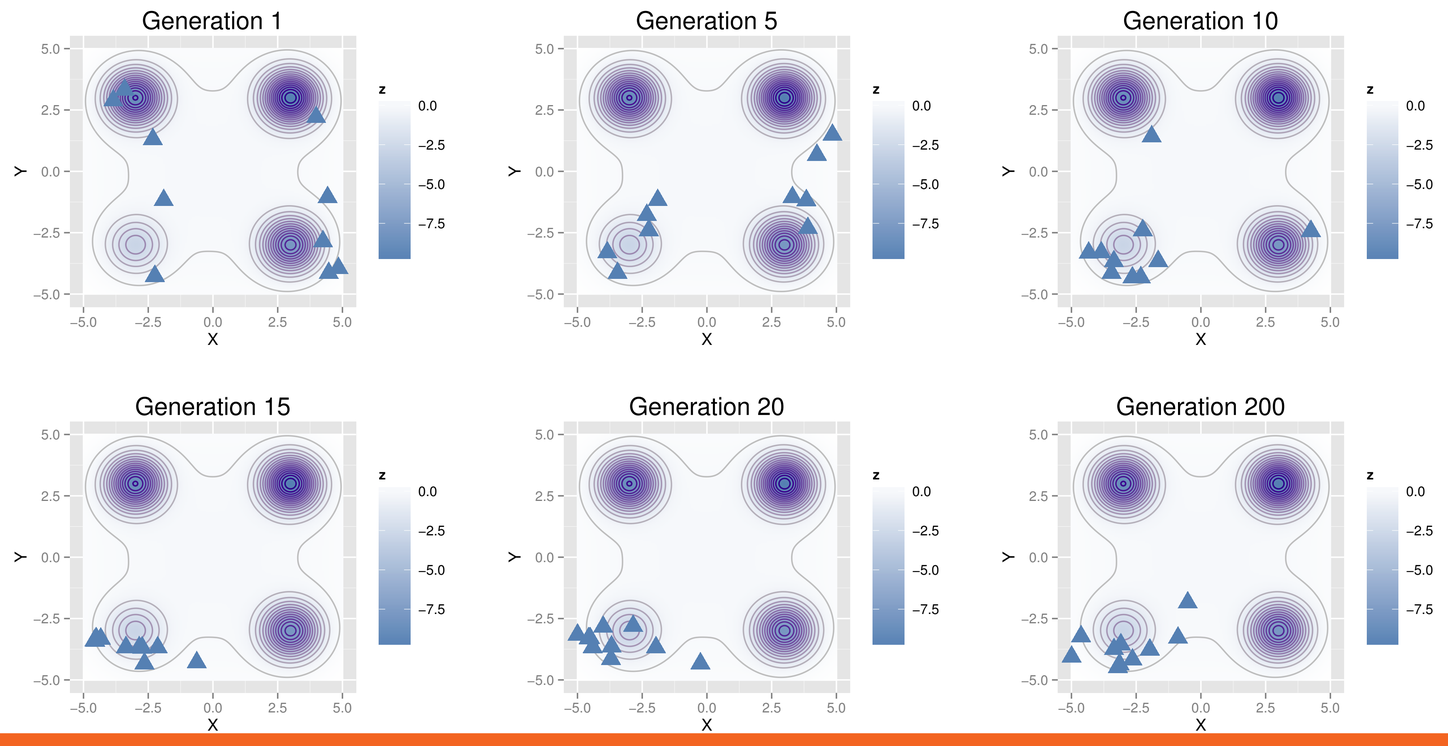
$$f_{\text{MultMod}}(x,y) = \frac{x^2+y^2}{100} - 10e^{-(x-3)^2-(y-3)^2} - 9e^{-(x+3)^2-(y-3)^2} - 3e^{-(x+3)^2-(y+3)^2} - 8e^{-(x-3)^2-(y+3)^2}$$

This function is shown in the following plot with signs changed.



Results

The following plot shows the movement of the prey individuals towards the worst starting positions.



The predator was considered as parameters for simulated annealing and BFGS. This example clearly portrayed the coevolutionary algorithms capabilities to find the hardest starting values, around the highest minimum. Where multiple optima are not present it learns to favor the best optimization algorithm for the job.

This poster only gives a brief overlook of the project. It also included tests on the `gadget` program, but there one relative fitness assessment can take from a few seconds to several minutes on small models. The scalability of the algorithm made parallelizing it easy.

All the code was written in R and `gadget` is written in C++. I want to thank the Marine Research Institute of Iceland for funding me while working on my thesis.

References

- Glover, Fred. *Future paths for integer programming and links to artificial intelligence.* Computers Operations Research 13, no. 5 (1986): 533-549.
- Samothrakis, Spyridon, Simon Lucas, Thomas Philip Runarsson, and David Robles. *Coevolving game-playing agents: Measuring performance and intransitivities.* Evolutionary Computation, IEEE Transactions on 17, no. 2 (2013): 213-226.
- Hillis, W. Daniel. *Co-evolving parasites improve simulated evolution as an optimization procedure.* Physica D: Nonlinear Phenomena 42, no. 1 (1990): 228-234.
- Einarsson, Guðmundur. *Competitive Coevolution in Problem Design and Metaheuristical Parameter Tuning*. M.Sc. Thesis. University of Iceland, Iceland (2014).
- Storn, Rainer, and Kenneth Price. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces.* Berkeley: ICSI, 1995.