CF967 - Notes on Co-evolution

Steve Phelps

March 9, 2010

1 Background Reading

1.1 Text books

- [Eiben and Smith, 2007, pp. 221–226]

1.2 Papers

- [Hillis, 1992]
- [Angeline and Pollack, 1993]

1.3 Third-party notes

- Introductory Tutorial on Coevolution
- Slides on Hillis' sorting networks

2 Overview

2.1 Definition

When an evolutionary algorithm has a fitness function which is a function of more than one phenotype we say that is *co-evolutionary*. Thus in a co-evolutionary algorithm fitness calculations are assessed from *joint interactions* between different members of the same or alternative populations. The design of co-evolutionary algorithms is biologically-inspired.

2.2 Evolutionary arms-races

Co-evolutionary algorithms were first introduced as a method for escaping from *local optima* in evolutionary computing when applied to optimisation problems. [Hillis, 1992] used this method for evolving sorting networks. He found that using a standard genetic algorithm, the evolutionary search became trapped in local optima. He then divided up the population into two separate sub-populations consisting of sorting networks (solutions) in one population, and sequences of numbers to be sorted (test cases) in the other population. The fitness of members of each population was assessed jointly. The sorting networks were assessed on the speed with which they sorted sequences of numbers, whereas the test-cases were assessed on the inverse objective, i.e. test-cases that challenged the sorting-networks and resulted in longer sort times were given high fitness. The idea was to try and stimulate an evolutionary arms race between the two opposing populations, thus driving the system out of local optima.

2.3 Relative fitness

The fitness of any individual depends on the composition of the population(s) at the time of assessment. Thus there is not a single static fitness landscape, since the fitness of an individual is assessed *relative* to other individuals.

This can make it hard to assess how well a co-evolutionary algorithm is progressing. In a non-coevolutionary algorithm we can plot the mean fitness of the population over the number of generations and straightforwardly determine whether the fitness is increasing, whereas such a graph would be meaningless in the case of a co-evolutionary algorithm.

3 Types of co-evolution

3.1 Single verses multiple populations

Different population structures can be used depending on the problem or modelling approach. Single-population models are often used to model social learning processes. In the context of modelling financial markets, this is the approach taken by [Lebaron and Yamamoto, 2007].

On the other hand, algorithms with multiple sub-populations are used when either: i) we have different types of agent with incompatible phenotypes; or (ii) we want to model interactions in which agents' strategies are not observable. For example, in [Hillis, 1992] there are two types of "agent": test-cases and sorting-networks. The phenotype for a sorting-network cannot be used to instantiate a test-case, thus it would be inappropriate to cross-over solutions from a single population. Thus [Hillis, 1992] uses two sub-populations corresponding to each *type* of solution. In strategic environments this would correspond to situations where a good strategy for one player would not necessarily be a good strategy for another player, i.e. assymetric games. In these models each sub-population is used to evolve the strategy for a particular player. [LeBaron et al., 1999] used a co-evolutionary model with multiple-subpopulations of rules for each agent.

3.2 Random verses full mixing

Fitness assessments can be conducted by a random tournament in which a random subset of agents is selected from the population(s) for each fitness evaluation. A special-case is random-pairing which is the implicit model used in evolutionary game-theory [Weibull, 1997, Gintis, 2000]. Alternatively all members of the population may be required to interact with every other individual before a fitness assessment is made ("full mixing").

3.3 Competitive verses Cooperative

The fitness function in a co-evolutionary algorithm is not necessarily zero-sum. Thus co-evolutionary algorithms can be used with positive-sum fitness functions which encourage mutalism. This latter situation is sometimes called "cooperative co-evolution" and can be used where we want to evolve separate sub-solutions to a problem which cooperate well together, for example separate control algorithms for different joints in an autonomous robot. See [Eiben and Smith, 2007, pp. 222-225] for a full discussion.

4 Pathologies

Despite some early successes there are several commonly-cited problems with co-evolutionary algorithms [Ficici and Pollack, 1998].

4.1 Mediocre-Stable States

Often when we attempt to induce arms-races between different sub-populations the system can reach a state where each sub-population "colludes" with the other and the population dynamics become stationary:

> *"...mediocre stable-states (MSS) are a common result in coevolutionary systems, where the agents in the evolving population(s), to anthropomorphise, discover a way to collude to give the impression of competition without actually forcing each other to improve in any objective sense. This phenomenon is analogous to that found in accounts of World-War I trench warfare, where opposing forces established ritualized acts of aggression meant to appear genuine to their respective commanders that were, nevertheless, completely predictable to each other, and thus of no real threat." [Ficici and Pollack, 1998, p. 1]*

4.2 Cycling

Sometimes the current population "forgets" how to beat members of earlier populations who have since died out. If these original solutions are rediscovered then the algorithm will cycle through phase-space.

4.3 Disengagement

Progress in a co-evolutionary algorithm relies on the competing solutions to maintain a gentle "fitness gradient". For example, in [Hillis, 1992] the test-cases should induce some fitness differential between the current sorting-networks. If the test-cases are too hard then all solutions will score equally poorly- thus the fitness gradient will be flat (uniform) and there will be no selection pressure to drive the sorting networks to do better.

4.4 Overspecialization

5 Countermeasures

There are two broad approaches to attempting to combat these “pathologies”.

5.1 Visualisation and Steering

The first approach is to differentiate between relative fitness and some hypothetical notion of objective performance from the perspective of the designer standing outside of the system. Of course, if we knew apriori how to measure objective performance then we would just use a standard evolutionary algorithm with the proposed metric as our fitness function. One way around this is to try and use scientific visualisation to study the dynamics of the co-evolutionary process. This may allow the designer to qualitatively assess whether the system is progressing in the intended way. [Cliff and Miller, 2006] introduce a method called CIAO plots for visualising progress (or lack of) in co-evolutionary algorithms. If visualisation is able to suggest to the designer that the co-evolutionary algorithm is not progressing correctly, this suggests that it might be fruitful to intervene at run-time in order to “steer” the system toward the intended behaviour, as suggested by [Bullock et al.,].

5.2 Pareto co-evolution and game-theoretic approaches

The second approach to dealing with these “pathologies” is to take relative fitness as the objective criteria. This leads to a game-theoretic understanding of co-evolutionary algorithms in which conditions such as “mediocre” stable states are viewed as *evolutionary* stable states; that is, the these states are not pathological but are solutions to a model which incorporates strategic-interaction.

[Noble and Watson, 2001] suggests using pareto coevolution to eliminate dominated strategies from the population(s). Think of this as a multi-objective optimisation in which the pareto-frontier consists of undominated strategies. [Ficici and Pollack, 2000] suggests comparing coevolutionary algorithms to the replicator dynamics, leading to a proposed algorithm for finding approximations of game-theoretic equilibria [Ficici and Pollack, 2003].

6 Co-evolution for agent-based modelling

When we use co-evolutionary algorithms in agent-based modelling, we are typically attempting to characterise the steady-state behaviour of a system comprised of adaptive agents. This contrasts with arms-race view of co-evolution as a means of escaping local optima when solving conventional optimisation-problems. Thus we typically take the game-theoretic approach as outlined in Section 5.2.

One way to think about this kind of co-evolutionary algorithm is as an approximate means of analysing multi-agent interactions when the space of possible pure strategies is too large to analyse game-theoretically. If the number of pure strategies is sufficiently small then we can use evolutionary game-theory to analyse the steady-state behaviour of the system. This is advantageous since we can use analytical methods to find the equilibria and characterise their stability. However, if we want to consider larger spaces of strategy we turn to heuristic search methods. Co-evolutionary algorithms are to game-theory as heuristic search is to mathematical optimisation. This does not mean, however, that we should throw away all the concepts from evolutionary game-theory when we perform an analysis using an agent-based model with co-evolutionary adaptation. Understanding the dynamics of learning is critical in both cases, thus we should attempt to understand and analyse the dynamics of learning in these systems. Plotting the evolution of agents in phase-space is just as valuable in an agent-based model as it is a mathematical biology model.

References

[Angeline and Pollack, 1993] Angeline, P. J. and Pollack, J. B. (1993). Competitive environments evolve better solutions for complex tasks. In Forrest, S., editor, *Genetic Algorithms: Proceedings of the Fifth International Conference (GA93).*

[Bullock et al.,] Bullock, S., Cartlidge, J., and Thompson, M. Prospects for computational steering of evolutionary computation. In *Proccedings of the Eighth Conference on Artificial Life.*

[Cliff and Miller, 2006] Cliff, D. and Miller, G. F. (2006). Visualizing coevolution with CIAO plots. *Artificial life*, 12(2):199–202.

[Eiben and Smith, 2007] Eiben, A. E. and Smith, J. E. (2007). *Introduction to Evolutionary Computing.* Natural Computing. Springer, Berlin, second edition.

[Ficici and Pollack, 1998] Ficici, S. G. and Pollack, J. B. (1998). Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of of ALIFE-6.*

[Ficici and Pollack, 2000] Ficici, S. G. and Pollack, J. B. (2000). A game-theoretic approach to the simple coevolutionary algorithm. In Marc Schoenauer, Kalyanmoy D., editor, *Parallel Problem Solving from Nature — PPSN VI 6th International Conference*, Paris, France. Springer Verlag.

[Ficici and Pollack, 2003] Ficici, S. G. and Pollack, J. B. (2003). A game-theoretic memory mechanism for coevolution. In Al, E., editor, *LNCS 2723*, pages 286–297. Springer-Verlag.

[Gintis, 2000] Gintis, H. (2000). *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction.* Princeton University Press.

[Hillis, 1992] Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Al, E., editor, *Proceedings of ALIFE-2*, pages 313–324. Addison Wesley.

[LeBaron et al., 1999] LeBaron, B., Arthur, W. B., and Palmer, R. (1999). Time series properties of an artificial stock market. *Journal of Economic Dynamics and Control*, 23(9-10):1487–1516.

[Lebaron and Yamamoto, 2007] Lebaron, B. and Yamamoto, R. (2007). Long-memory in an order-driven market. *Physica A: Statistical Mechanics and its Applications*, 383(1):85–89.

[Noble and Watson, 2001] Noble, J. and Watson, R. A. (2001). Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 493–50, San Fancisco, California. Morgan Kauffman.

[Weibull, 1997] Weibull, J. W. (1997). *Evolutionary Game Theory.* MIT Press, first mit press edition.