An Evolutionary Approach to Complex System Regulation Using Grammatical Evolution

Saoirse Amarteifio and Michael O'Neill
Biocomputing and Developmental Systems
University Limerick
Ireland
amartey@eircom.net, michael.oneill@ul.ie

Abstract

Motivated by difficulties in engineering adaptive distributed systems, we consider a method to evolve cooperation in swarms to model dynamical systems. We present an information processing swarm model that we find to be useful in studying control methods for adaptive distributed systems. We attempt to evolve systems that form consistent patterns through the interaction of constituent agents or particles. This model considers artificial ants as walking sensors in an information-rich environment. Grammatical Evolution is combined with this swarming model as we evolve an ant's response to information. The fitness of the swarm depends on information processing by individual ants, which should lead to appropriate macroscopic spatial and/or temporal patterns. We discuss three primary issues, which are tractability, representation and fitness evaluation of dynamical systems and show how Grammatical Evolution supports a promising approach to addressing these concerns.

Introduction

Nature clearly sets the standard on complex system regulation. However her principles can be difficult to apply forcing us to ask difficult questions about how to allow open-ended evolvability, how to reach high levels of adaptability or how to regulate developmental processes. We take an *evolutionary automatic programming* approach in addressing some early issues in using natural principles to program artificial complex systems. Specifically we begin by considering how synergy in swarms can be evolved to produce consistent patterns from the interaction of simple agents. Using Grammatical Evolution (O'Neill and Ryan, 2003) we evolve templates for simple transducers that describe how environmental information should be modelled by ants to produce responses that favour a swarm's fitness.

Background to evolving adaptive behaviour is outlined in section 2 and we discuss issues in engineering distributed adaptive systems. We provide background on grammatical evolution in section 3. In Section 4 we describe information processing swarms as an interesting model for regulating complex dynamical systems and describe a new model for evolving swarms. We outline three issues which we consider crucial; tractability, representation, and fitness evaluation. Section 5 describes the swarm model. Results comparing two experiments can be found in section 6. We close with a discussion of results.

Background

Swarm Intelligence, which is comprised of Particle Swarm (Kennedy and Eberhart, 1995; Kennedy and Eberhart, 2001) and Ant Algorithms (Colorini et al., 1991; Dorigo et al., 1996; Bonabeau et al., 1999), has the potential to be used as a model for regulating distributed adaptive systems. It has been said in the context of dynamical systems and morphogenesis that finding a solution to a particular [distributed] problem is equivalent to finding a specific pattern in space or time (Bonabeau, 1997). It is of interest to understand how patterns unfold through random processes and interactions among elementary constituents or agents (Bonabeau, 1997). Swarms of interacting, information processing agents can yield the emergence of a computational power not present at the level of individual organisms. Examples of artificial systems that possess this computational power are dynamical systems such as Swarms and Cellular Automata (CA) (Wolfram, 1983; Crutchfield et al., 2003). Dynamical systems are often considered cognitive or purposeful and considered in light of achievements such as crop harvesting in ant colonies or pathogen detection in the immune system. The current study is interested in the system regulation *per se*, considering these achievements or patterns simply as by-products of system dynamics.

Dynamical systems are characterised by a continual coupling between systems and their environment. They are spatially extended, consisting of a large number of simple components each with limited communication to other components and each following simple transition rules that depend on their inputs. Properties of such systems although attractive are difficult to formalize for engineering purposes (Rosen, 1985; Kubrik, 2003). These principles are likely to be of great importance to adaptive distributed systems design over the coming years in fields such as amorphous computing (Abelson et al., 2000; Nagpal, 2001). Researchers in [illegible] prefer to understand the construction of these sys-

tems so that they function as intended not as they evolve. In support of the direct engineering approach, (Nagpal, 2002; Nagpal, 2001) observes that in CA research, local rules are constructed empirically without providing a framework for construction of local rules to obtain any desired goal and that these approaches are difficult to generalize. On the other hand Nagpal observes that evolutionary computing, while generalising well, uses local rules that are evolved without any understanding of how they work. This problem is compounded when evolving dynamical systems as the emergent computation performed is determined by space-time behaviour (Crutchfield et al., 2003). Designing an appropriate fitness function can be as difficult as designing a control algorithm from scratch (Nagpal, 2001).

Yet we ask how evolution can be used to produce a better learning system, which could be called adaptive, dynamical or emergent. Similar work has been done in the fields of Artificial Neural Networks, which has been termed Evolutionary Artificial Neural Networks (EANN) (e.g. see (Yao, 1999)), evolving Cellular Automata (e.g. see (Crutchfield et al., 2003)). Very recently similar work has been carried out in the field of Swarm Intelligence (Williams, 2002) and the evolution of mulitcelluar programs using genetic programming (Schmutter, 2002).

Our work bears only principle similarities of varying degrees with past work in EANNs and evolving CA. CA are more akin to swarms as both are dynamical systems unlike ANNs, which are computational and have different properties. We are interested in evolving dynamical systems which are characterized by continual change, have less imposition of structure giving rise to an enabling substrate whereby higher level functionality can emerge. See (Mitchell, 1998) for discussion on dynamical versus computational systems. We deliberately avoid discussion of cooperation and the evolution of cooperation stemming from Robert Axelrod's work (Axelrod, 1987) as this work is largely concerned with cooperation among self-interested agents.

Grammatical Evolution

Grammatical Evolution (GE) is an evolutionary algorithm that can evolve computer programs in any language (O'Neill and Ryan, 2003; O'Neill, 2001; O'Neill and Ryan, 2001; Ryan et al., 1998), and can be considered a form of grammar-based genetic programming. Rather than representing the programs as parse trees, as in GP (Koza, 1992; Koza, 1994; Banzhaf et al., 1998; Koza et al., 1999; Koza et al., 2003), a linear genome representation is used. A genotype-phenotype mapping is employed such that each individual's variable length binary string, contains in its codons (groups of 8 bits) the information to select production rules from a Backus Naur Form (BNF) grammar. The grammar allows the generation of programs in an arbitrary language that are guaranteed to be syntactically correct, and as such it is used as a generative grammar, as opposed to the classical use of grammars in compilers to check syntactic correctness of sentences. The user can tailor the grammar to produce solutions that are purely syntactically constrained, or they may incorporate domain knowledge by biasing the grammar to produce very specific forms of sentences.

BNF is a notation that represents a language in the form of production rules. It is comprised of a set of non-terminals that can be mapped to elements of the set of terminals (the primitive symbols that can be used to construct the output program or sentence(s)), according to the production rules. A simple example BNF grammar is given below, where `<expr>` is the start symbol from which all programs are generated. These productions state that `<expr>` can be replaced with either one of `<expr><op><expr>` or `<var>`. An `<op>` can become either `+`, `-`, or `*`, and a `<var>` can become either `x`, or `y`.

```
<expr> ::= <expr><op><expr>  (0)
         | <var>             (1)
  <op> ::= +                 (0)
         | -                 (1)
         | *                 (2)
 <var> ::= x                 (0)
         | y                 (1)
```

The grammar is used in a developmental process to construct a program by applying production rules, selected by the genome, beginning from the start symbol of the grammar. In order to select a production rule in GE, the next codon value on the genome is read, interpreted, and placed in the following formula:

$$Rule = Codon\ Value\ \%\ Num.\ Rules$$

where % represents the modulus operator.

Beginning from the the left hand side of the genome, codon integer values are generated and used to select appropriate rules for the left-most non-terminal in the developing program from the BNF grammar, until one of the following situations arise: (a) A complete program is generated. This occurs when all the non-terminals in the expression being mapped are transformed into elements from the terminal set of the BNF grammar. (b) The end of the genome is reached, in which case the *wrapping* operator is invoked. This results in the return of the genome reading frame to the left hand side of the genome once again. The reading of codons will then continue unless an upper threshold representing the maximum number of wrapping events has occurred during this individuals mapping process. (c) In the event that a threshold on the number of wrapping events has occurred and the individual is still incompletely mapped, the mapping process is halted, and the individual assigned the lowest possible fitness value. A full description of GE can be found in (O'Neill and Ryan, 2003).

Swarm Evolution

We consider three primary concerns in evolving dynamical systems.

Tractability The problem tends to be intractable in terms of evolutionary time and search. Emergent computation performed by CA is determined by its overall space-time behavior (Crutchfield et al., 2003). This is also true of swarms. Individual experiments evaluated by the evolutionary system may run for several minutes. The problem contains non-linearities and therefore mirrors a rugged search space. On a desktop PC it could take weeks to evolve a solution, if a solution is to be found at all.

Representation As an automatic programming problem we must consider how to encode the problem, what terminals to use and how to map the genetic material onto the swarm program. This representation should contribute to a Language that allows us to relate microscopic and macroscopic phenomena (Kubrik, 2003).

Fitness Evaluation A Method to determine the fitness of the swarm can be difficult to produce for dynamical systems problems (Williams, 2002). Methods to identify patterns in space or time are required. For example Spatial Entropy values (Bonabeau et al., 1999), Hough Transforms (Williams, 2002) and techniques from computational mechanics (Crutchfield et al., 2003) each describe patterns and hence fitness of dynamical pattern-forming systems. This is not to say there is a one-size-fits-all fitness evaluation method for complex systems but that dynamical systems might be evaluated based on the recognition of patterns.

In previous work we ran a number of experiments that considered the information processing capability of swarms on clustering tasks and considered the impact of different information usage. We concluded that different functions of information did effect the swarm's ability to model its environment although it was difficult to determine appropriate information and functions. Each dynamical environment has its own properties and information - where information is the meaning of events. In complex environments there are consistent patterns in space and time. If an entity can recognise relevant patterns it has the potential to be adaptive and anticipatory (Rosen, 1985). The information-theoretic notion of relevant information and the potential of taking an agent-centered theoretic approach to the design of distributed adaptive systems is discussed in (Nehaniv et al., 2002).

Every creature in nature survives by modelling its environment as it senses it and through a transducer system makes a response that is fit for that environment. Using GE, we evolve templates for simple transducers that describe how environmental information should be modelled by an ant to produce a response that favours the swarm's fitness.

Many of the distributed systems envisioned are data and event-centric and closely tied to their environments. Agents will make appropriate responses in the presence of certain environmental information. Grammar-based Genetic Programming approaches such as GE are a powerful means to describe legal interpretations of such terminal information yet still allow the open-ended evolution of novel solutions. GE's distinction between genotype and phenotype aids in representation of the problem and provides a substrate for processes that will regulate swarm construction.

Experimental Setup

We describe a multi-agent or swarm simulation model where information processing ants cooperate to solve an abstract clustering problem. Ants cluster identical objects which is a distinct problem from sorting (where a similarity measure is used to sort like objects). The ant's world is a square toroidal grid of 177*177 pixels (similar area to the circular world used in (Bonabeau et al., 1999))and uses a Moore neighbourhood. We use the Repast simulation tool and the Colt math library (Repast, 2004). Ants will move one pixel per time step. Ants can move in random directions based on a Gaussian probability distribution centered around the forward direction. A 'left antenna' at the north-west Moore pixel surrounding the ant and a 'right antenna' at the north-east are used to sense the concentration gradient. An ant will move deterministically in the direction of highest concentration or will continue moving straight if there is no difference in concentrations. These design choices are based on natural phenomena (Wilson, 1971; Hölldobler and Wilson, 1990). An ant will deposit a single pheromone signal of a certain concentration as it moves. This pheromone signal diffuses and evaporates at a constant rate. Diffusion and evaporation is implemented by the repast simulation tool (Repast, 2004). An ant may pickup or deposit objects in it's environment. Ants have a 'browsing' function whereby objects and other ants are observed and 'remembered'. The ant has a limited memory map where each object type encounter over a certain period is stored.

Ants use non-deterministic threshold functions (Bonabeau et al., 1999) to determine action. Equations 1 and 2 give the probability to pick up and drop an object respectively. $k1$ is the threshold value for picking up and object and $k2$ is the threshold value for dropping an object. f is the stimulus that the ant perceives - here it is the fraction of objects perceived over some period i.e. the ant's memory length.

$$\rho_p = \left(\frac{k1}{k1+f}\right)^2 \quad (1)$$

$$\rho_d = \left(\frac{k2}{k2+f}\right)^2 \quad (2)$$

The parameters used in the simulation model are shown in Table 1. We experimentally decided the simulation time.

Parameter	Value
World Dimensions	177 X 177
Ant Count	400
Object Count	1000
Memory Length	50
Decay factor	80 timesteps
Evaporation Rate	0.87
Diffusion Rate	0.42
Simulation Duration	5000 timesteps

Table 1: Swarm simulation parameters.

This time is a short duration that can be used to effectively evaluate the clustering performance. Evaporation and diffusion rates were chosen to represent 'recruitment' signals with medium spatial effect and short temporal effect. Other parameters were chosen over a number of trials. In the clustering task, ants have a number of behaviours. There is a behaviour for picking up objects, dropping objects, sensing stimulus, dropping pheromone and depositing 'pheromone traces' on objects. Each of these has a corresponding *gene*, which is the evolved template mentioned above. These templates are equivalent to GP S-Expressions, arranged in sequences that make up a complex of S-Expressions or genes. These genes regulate ant responses using environmental information as inputs. Each ant individual is encoded by a complex of genes mapped to behaviours. These genes regulate the values used for k1, k2 and f in equations 1 and 2. When objects are deposited, they emit pheromone traces of concentration specified by the appropriate gene. The concentration of pheromone used in ant trails is also regulated by a gene. Pheromone is emitted for a period of time specified by the decay factor in Table 1. GE uses the following grammar to map a genome to an S-Expression, or a complex of S-Expressions called GeneComplex. Start Symbol, S, Non-terminals, N, Production rules, P, and Terminals, T are shown below.

```
S = GeneComplex
P = GeneComplex> ::=    <expr> <expr>
                        <expr> <expr>
                        <expr>   (0)

<expr> ::= <expr> <op> <expr> (0)
         | <var>              (1)

<op> ::=  + (0)
        | - (1)
        | / (2)
        | * (3)

<var> ::= 10                  (0)
        | 100                 (1)
        | ants                (2)
        | working_ants        (3)
        | current_pheromone   (4)
        | pheromone           (5)
        | objects             (6)
```

Notice the terminal symbols of the grammar correspond to environmental information. Production rule 1 describes a complex of 5 S-Expressions. These expressions are mapped onto ant behaviours. A complex of S-Expressions are mapped in an arbitrary but consistent manner from a single genome using the grammar above. This complex is passed to a swarm simulation and used to construct genes in ants using a sequential mapping of S-Expressions onto the behaviour to be encoded. During the simulation at each time step, the ant senses its environment and passes in a table of all sensed variables corresponding to terminal symbols. Terminal symbols represent the number of ants, working ants, objects etc. (see grammar) encountered at each time step. Each behaviour class then computes a response value based on the environmental information supplied. These computed values are used for example in equations 1 and 2 to decide when to pick up and drop objects. In this way, each S-expression determines the value for a gene or parameter that determines the expression of a behaviour based on information inputs.

After the specified experiment length, the fitness of the swarm is computed and returned to the evolutionary engine. Spatial Entropy (Gutowiz, 1993; Bonabeau et al., 1999) was used as a measure for clustering performance in a lattice divided into a number of grids. The equation below gives the spatial entropy E, at a certain grid scale, s. P is the fraction of objects in one grid of total objects. In our experiments s = 6, there are 36 grids.

$$E_s = -\sum_{I \in s-patches} P_I log P_I \qquad (3)$$

Spatial Entropy (Gutowiz, 1993) is a macroscopic measure that corresponds to the individual ant's microscopic goals. As the ants 'work', spatial entropy values tend to decrease as the world becomes more ordered. In time-dependant problems it is advantageous to have fitness evaluation measures where the fitness value tends towards the optimum. In previous work we ran experiments for 50,000 time steps. This was a reasonable 'convergence' time over different clustering models. As mentioned, we run simulations for only 5,000. However, this time reasonably approximates clustering behaviour over 50,000 time steps.

It is the nature of the swarm clustering task that it should be easy to find a good solution (approximate ratio between stimulus and threshold values) while it can be difficult to identify the features of the multi-parameter problem. Consequently we focus on showing evidence of progressive search rather than on finding an optimal solution. GE genome individuals are evaluated only once (assuming they are not mutated). We use a variable-length generational GA with tournament selection, one-point crossover and integer mutation (as opposed to bit mutation). See evolutionary parameters table 2

Parameter	Value
Mutation rate	.1
Crossover Rate	.7
Population size	200
Generations	5
Fitness Measure	Spatial Entropy

Table 2: GE parameters.

Generations	1	2	3	4	5
GE(mean)	1.79	1.79	1.70	1.43	1.70
GE(sd)	0.47	0.11	0.11	0.35	0.33
Rand(mean)	2.02	1.99	2.42	1.65	1.98
Rand(sd)	0.60	0.45	0.49	0.47	0.15

Table 3: Comparison between random run (Rand) and Grammatical Evolution (GE) showing mean Spatial Entropy results and standard deviation. (Initial entropy values approx. 2.7 on average)

Results

Results showing best solutions found in each generation for both GE search and random search are shown in Table 3. Random search generates and evaluates a random population on each generation. Due to computational time requirements, we have only taken 5 samples of each. We hoped to observe evidence of evolvability (Altenberg, 1994) rather than find optimal solutions. In evolving complex systems it would seem more important to enable *scaffolding*, where we maintain good sub-structures in a population and build on them. Although not discernable from the tables above, GE did find the most favourable spatial entropy value of 1.059 although only marginally better than the best value in the random search which was 1.127. However GE made improvements over successive generations in most samples. Over many clustering experiments, spatial entropy values ranged from average 2.7 (worst) to 0.9 (best) using given parameters.

We observed a number of clustering patterns. All experiments used the same agent models and differed only in the information-processing templates used. Clustering models in the literature show consistent formation of several small clusters, gradually becoming three or four large clusters. We observed these similar patterns but also observed patterns where objects seemed to be 'swept' into regions of the ant's world. The regions first contained sparse clusters that were gradually swept into dense compact clusters. We observed clusters that formed stripe-like patterns in addition to 'spots' although this may have been as a result of agent trajectories.

Conclusions

The purpose of this article was to demonstrate evolutionary pattern-forming swarms using Grammatical Evolution, with the result that the ant colony successfully evolved templates that exhibited clustering behaviour based on a spatial entropy measure. We have focused on the representation of evolving dynamical systems. We feel our information-theoretic representation could be easily generalized. GE provides independence between the evolutionary aspects and the program representation. Many complex systems can be considered in terms of swarms of information processing particles. The use of grammars provides a means to describe transducers for information processing in complex system nodes. Grammars provide a powerful means to describe legal interpretations of information yet still allow the open-ended evolution of novel solutions.

Fitness evaluation methods that evaluate patterns are an interesting way to evolve dynamical systems. The choice of fitness function is important as depending on how well it approximates or anticipates performance of the dynamical system in the early stages of a simulation, one can use shorter simulation times and less runs in the evolutionary stages.

The use of templates in a homogenous colony leads to behaviorally heterogenous ants based on their environmental information context. In a sense it also realises a type of ontogeny in that over time, the ant has features that may have variable fitness. This is an important aspect to exploit given the temporal development of simulations and makes the colony more adaptive. We have observed this through comparisons between models using static parameters and those using template-based parameters. In all areas of complex system research, a bridge of understanding between microscopic and macroscopic phenomena is required. Some research perspectives focus more on one or the other of these suffering the critique of others. Templates represent for us loci at which to study this connection. The evolutionary search implicitly identifies these templates as features of complex systems where for some global task, we can see how individuals process information. For example, even on the simple clustering task we observed several patterns of clustering. Given that we can observe these macroscopic patterns, can we analyze the templates and find out what the ants were thinking?

References

Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T. F., Nagpal, R., Rauch, E., Sussman, G. J., and Weiss, R. (2000). Amorphous computing. *Communications of the ACM*, 43(5):74–82.

Altenberg, L. (1994). The evolution of evolvability in genetic programming. In Kinnear, K. E., editor, *Advances in Genetic Programming*, pages 47–74. MIT Press, Cambridge, MA.

Axelrod, R. (1987). The evolution of strategies in the iterated prisoner's dilemma. In Davis, L., editor, *Ge-*

netic Algorithms and Simulated Annealing, pages 32–41. Princeton University Press.

Banzhaf, W., Nordin, P., Keller, R., and Francone, F. (1998). *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann.

Bonabeau, E. (1997). From classical models of morphogenesis to agent-based models of pattern formation. *Artificial Life 3*, pages 191–209.

Bonabeau, E., Theraulaz, G., and Dorigo, M. (1999). *Swarm Intelligence*. Oxford Press.

Colorini, A., Dorigo, M., and Maniezzo, V. (1991). Distributed optimsation by ant colonies. In Varela, F. and Bourgine, P., editors, *European Conference on Artifical Life*, pages 134–142. MIT-Press.

Crutchfield, J., Mitchell, M., and Das, R. (2003). Evolutionary design of collective computation in cellular automata. In Crutchfield, J. and Schuster, P., editors, *Evolutionary Dynamics*. Oxford University Press.

Dorigo, M., Maniezzo, V., and Colorini, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions*, pages 29–41.

Gutowiz, H. (1993). Complexity seeking ants: (Unpublished).

Hölldobler, B. and Wilson, E. O. (1990). *The Ants*. Sprinter-Verlag.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimzation. In *IEEE International Conference On Neural Networks*, pages 1942–1948.

Kennedy, J. and Eberhart, R. (2001). *Swarm Intelligence*. Morgan Kaufmann.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Koza, J. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press.

Koza, J., Andre, D., Bennett III, F., and Keane, M. (1999). *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufmann.

Koza, J., Keane, M., Streeter, M., Mydlowec, W., Yu, J., and Lanza, G. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers.

Kubrik, A. (2003). Towards a formalisation of emergence. *Artifical Life*, 9:41–65.

Mitchell, M. (1998). A complex-systems perspective on the 'computation vs. dynamics' debate in cognitive science. In *Twentieth Annual Conference of the Cognitive Science Society*.

Nagpal, R. (2001). *Progammable Self-Assembly: Constructing Global Shape using Biologically-Inspired Local Interactions and Oragami Mathematics*. PhD thesis, Massachusetts Institute of Technology.

Nagpal, R. (2002). Programmable self-assembly using biologically-inspired multiagent control. *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 418–425.

Nehaniv, C. L., Polani, D., and Dautenhahn, K. (2002). Meaningful information, sensor evolution, and the temporal horizon of embodied organisms. pages 345–349. MIT Press.

O'Neill, M. (2001). *Automatic Programming in an Arbitrary Language: Evolving Programs in Grammatical Evolution*. PhD thesis, University of Limerick.

O'Neill, M. and Ryan, C. (2001). Grammatical Evolution. *IEEE Transactions on Evolutionary Computation*.

O'Neill, M. and Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. KluwerAcademic Publishers.

Repast (2004). http://repast.sourceforge.net, Social Science Research Computing University of Chicago.

Rosen, R. (1985). *Anticipatory Systems: Philosophical, Mathematical and Methodological Foundations*. Pergamon Press.

Ryan, C., Collins, J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of the First European Workshop on Genetic Programming*, pages 83–95.

Schmutter, P. (2002). Object-oriented ontogenetic programming: Breeding computer programs that work like multicellual creatures. Master's thesis, University Dortmund.

Williams, H. (2002). Spatial organisation of a homogenous agent population using diffusive signalling and role diffrenciation. Master's thesis, University Sussex.

Wilson, E. O. (1971). *The Social Insects*. Harvard University Press.

Wolfram, S. (1983). Statistcal mechanics of cellular automata. *Review of Modern Physics*, pages 601–644.

Yao, X. (1999). Evolving artificial neural networks. In *Proceedings of the IEEE Vol 87 No 9*, pages 1423–1447.