Shaping collective behavior: an exploratory design approach

Pablo Funes[1], Belinda Orme[1] and Eric Bonabeau[1]

[1]Icosystem Corporation, 10 Fawcett Street, Cambridge, MA 02138, USA
{pablo, belinda, eric}@icosystem.com

Abstract

In order to fulfill the true promise of decentralized, self-organizing intelligence, a major design problem has to be overcome. Designing the individual-level rules of behavior and interaction that will produce a desired collective pattern in a group of human or non-human agents is difficult because the group's aggregate-level behavior may not be easy to predict or infer from the individuals' rules. While the forward mapping from micro-rules to macro-behavior in self-organizing systems can be reconstructed using computational modeling techniques such as agent-based modeling, the inverse problem of finding micro-rules that produce interesting macro-behavior poses significant challenges, all the more as what constitutes "interesting" macro-behavior may not be known ahead of time. An exploratory design method is described in this paper. It relies on interactive evolution. We show how it can be used to discover new, "interesting" patterns of collective behavior when one does not know in advance what the system is capable of doing, a generic situation in the design of collective intelligent systems.

Introduction

Designing the individual-level rules of behavior and interaction that will produce a desired collective pattern in a group of human or non-human agents is difficult because the group's aggregate-level behavior may not be easy to predict or infer from the individuals' rules. For example, the aggregate-level properties of traffic jams (Helbing *et al.*, 2000), a crowd evacuating a public space (Still, 1993) or the stock market (Palmer *et al.*, 1994) cannot easily be derived from knowing the rules of behavior and interaction of drivers, people or investors. Agent-based modeling (ABM) (Reynolds, 1987; Epstein & Axtell, 1996; Axelrod, 1997; Bonabeau, 2002), or micro-simulation, is often the only way to capture the emergent properties resulting from the behavior and interactions of the group's constituent units or agents, particularly but not only when the individual-level rules are discrete. While ABM is useful in producing aggregate-level patterns from individual-level rules, finding the appropriate rules still requires manual search and tinkering when (1) the collective-level patterns may be difficult to formalize into a mathematical detector and therefore the evaluation of a solution cannot be automated, and/or (2) the collective-level patterns made possible by the individual-level rules are not even known ahead of time. If the desired collective-level pattern is known and its detection can be automated, then traditional search and optimization algorithms can be utilized. This paper focuses on cases where (2) is true (but of course, (2) implies (1)). We show that in such cases a technique originally developed to generate "interesting" images and pieces of art (Dawkins, 1987; Sims, 1991, 1992, 1993) can be used to design the individual-level rules of behavior and interaction to produce "interesting" collective-level patterns.

The technique (see Takagi, 2001 for a review) is a directed search evolutionary algorithm which requires human input to evaluate the fitness of a collective-level pattern (here, the fitness might be how close the collective-level pattern is to the desired pattern, or how interesting the pattern is) and uses common evolutionary operators such as mutation and crossover (Goldberg, 1989; Forrest, 1993) to breed the individual-level rules that produced the fittest collective-level patterns. Using a simple example of a human game that can be played in small groups (Bonabeau, 2002; Funes *et al.*, 2003), we show that this approach is particularly powerful as an exploratory design technique, when the aggregate-level capabilities of the system are not known. Interactive evolutionary computation (IEC), as this technique is known, combines computational search with human evaluation (Takagi, 2001).

Some of the individual-level rules discovered using IEC are presented together with their corresponding striking, unexpected patterns. In itself, the discovery of the rules is important as it shows that it is possible to design simple rules to produce robust (genetic robustness being a by-product of the evolutionary method: to be discovered, a fitness peak has to be reasonably stable under a number mutations), collective-level patterns; in addition, the IEC technique used to discover these rules is very generic and makes it possible to systematically discover novel phenomena in self-organizing systems (Bonabeau *et al.*, 2000).

A Simple Game

To illustrate the approach, a game of aggressors and defenders is used with a group of N players. Two rules can be used by the players:

- Rule #1: Pick two people A and B in the group. Then start moving in such a way that B (your defender) is between yourself and A (your aggressor).

- Rule #2: Pick two people *A* and *B* in the group. Then start moving in such a way that you are between *A* (the aggressor) and *B* (the defendee).

If for example everyone in the group follows Rule #1 and picks *A* and *B* randomly, the resulting collective-level pattern is chaotic, sometimes room-filling motion across the room, often constrained by the walls, which leads to some wall following. If on the other hand everyone in the group follows Rule #2 and picks *A* and *B* randomly, the resulting collective-level pattern is the rapid formation of a cluster, that is, the whole group collapses onto a single cluster. These two simple versions of the game can easily be played with a group of 8 and more (our experience includes playing the game with up to 400 people) and produce aggregate-level patterns that cannot easily be predicted from an examination of the individual rules. For example, in both versions there is exactly the same number of aggressors, defenders, defendees. While it is difficult to predict the outcome of even the simplest versions of the game, Anderson (2003) has established elements of a mathematical proof for the two situations described above (all participants following the same rule and picking their *A*'s and *B*'s randomly). A complementary approach consists of simulating the mapping from micro-rules to macro-behavior through agent-based modeling (ABM) (Epstein & Axtell, 1996; Bonabeau, 2002). Figure 1 sketches how Rule #1 and Rule #2 are represented in the model.

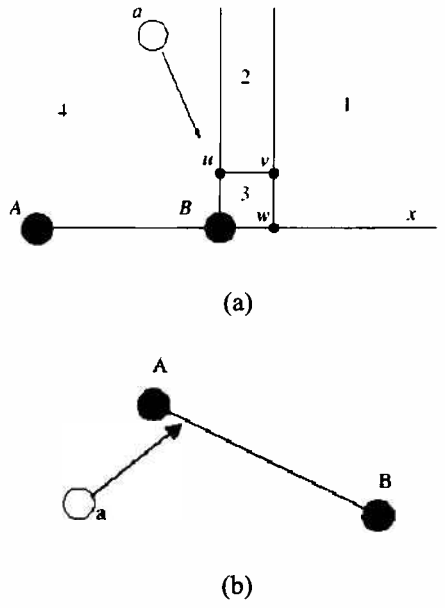

Figure 1: Definition of Micro-rules #1 (a) and #2 (b) in the agent-based model. (a). The goal for agent *a* is to have *B* protecting it from *A*. Depending on the relative positions of *A* and *B*, it proceeds as follows: in region 1 a moves toward the closest point in line *x*. In region 2, *a* moves toward point *v*. In region 3, a moves toward point *w*. In region 4, *a* moves toward point *u*. Points *u*, *v* and *w* are defined as follows: let *r* be the radius of one agent, then the distance between *u* and *B*, *u* and *v*, *v* and *w* is equal to 2*r*. The rule is satisfied, and the agent stands still, anywhere over the line *x* and to the right of *w*. Symmetrical regions (not shown) exist below *x*. (b) *A* similar approach was used to define the defender rule. *a* tries to satisfy the defender rule "place yourself between person *A* and person *B*" by moving toward the closest point to its current position on the line from *A* to *B*.

Based on the angle of motion θ_i calculated as indicated in Figure 1, the position A_i, represented as a complex number, of agent *i* is updated:

$$A_i \leftarrow A_i + \sigma e^{i\theta_i}, \qquad (1)$$

where σ represents speed, unless there is a collision with a wall or another agent, in which case σ is decreased so as to avoid the collision. Discrete time steps are used in the simulation, with $\sigma = \delta$ step^{-1}, where δ is the distance covered in one time step; here $\delta=2r$. Figure 2 illustrates the patterns observed in the two simple cases described above when simulated using agent-based modeling.

In some situations other than the two simple ones described above, we discovered by playing the game live that the aggregate-level pattern could be quite different from the above-mentioned patterns. For example, if by chance everyone picks the same person as aggressor or defender of defendee, the resulting aggregate-level pattern is very different. In other situations the relationship graph has several disconnected components (although the probability of observing more than one component if participants pick A and B tends uniformly randomly toward 0 as N^3 (Anderson, 2003)), leading to the formation of several clusters.

After these phenomena were observed *in vivo*, the corresponding micro-rules were simulated using ABM and could be re-created *in silico*, proving the predictive power of ABM. The observation of these unexpected collective patterns of behavior prompted us to ask the following question: would it be possible to design social networks (characterized by relationship graphs: who interacts with whom and what is the nature of the interaction?), instead of creating random ones, to produce interesting aggregate-level patterns, with no *a priori* knowledge of what interesting patterns this system could produce? More precisely, every individual is characterized by the following set of rules or properties:

- Does the individual follow Rule #1 or Rule #2?
- Who is *A*?
- Who is *B*?

The size of the space of relationship graphs (which we will also call rule space) grows fast with the number of individuals *N*. While rule space is large, it is likely that most resulting collective patterns will either be random-looking or reducible to one of the two basic patterns described above (chaotic behavior or clustering).

However we do not know what such a system can or cannot do, that is, we do not know what kind of collective-level patterns to expect other than chaotic motion and clustering. In Section 3 we describe a method to search for the relationship graphs, or individual-level rules, that will produce the most interesting collective-level patterns with only a vague notion of what interesting means: neither random nor reducible to one of the two basic patterns.

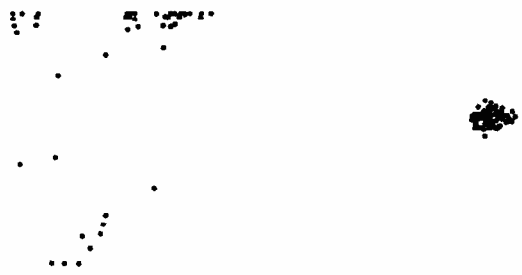

Figure 2: Micro-rules #1 (a) and #2 (b) lead to two dramatically different collective behavior in reality and in the predictive agent-based simulation.

Interactive evolution

The IEC search method works as follows:

1. A small initial population of relationship graphs is generated.
2. The resulting collective-level dynamical patterns are generated using an agent-based simulation and presented to a human observer.
3. The observer selects the collective-level patterns that are the most interesting –the fittest individuals in the population according to whatever set of objective and subjective criteria the observer may be using.
4. A new population (new generation) of relationship graphs is generated by applying mutation and crossover operators to the relationship graphs that correspond to the previous generation's fittest collective-level patterns (Goldberg, 1989; Forrest, 1993). In addition to the offspring and mutated versions of those graphs, randomly generated graphs are injected into the population to ensure diversity.
5. The new population is then simulated and the resulting collective-level patterns presented to the observer, and so forth.
6. This procedure is iterated until interesting patterns emerge from the search.

The user interface, which is a critical component of the method as it is based on visualizing the solution that the observer evaluates solutions, is shown in Figure 3. Obviously this method can only work if the population size is kept small and if interesting patterns emerge after a reasonably small number of generations.

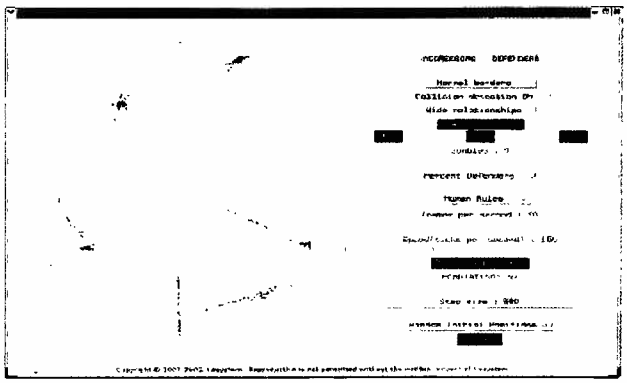

Figure 3: User interface with six playgrounds. Clicking on the playground that displays the preferred behavior results in the remaining ones becoming mutants or recombinations of the selected one.

Details of the evolutionary algorithm

The notation we use to pertain to a given generation is the subscript notation such that game i in generation n and $n+1$ is given by i_n and i_{n+1} respectively.

Selection of the fittest game and elitism

The game designated as the fittest by the user in generation n, j_n, is selected for preservation in generation $n+1$. The remaining games in generation $n+1$, i_{n+1}, will consist of agents derived from either a recombination or a mutation of some the agents in the original game, j_n. This implies mutation and recombination occur with a probability of 0.5. For example if population size is 9 games, one will be identical to the fittest game selected in the previous generation; four will have some agents in their population altered by recombination; and four will have some agents in their population altered by mutation.

Recombination.

If the rules for game i_{n+1} are to be created from a recombination of the rules describing agents in game j_n, then a uniformly random fraction drawn between 0% and 10% of the total number of agents in game i_{n+1} are derived in the following manner: a typical agent selected to be changed is the target agent, p; another source agent, q, is chosen randomly from the population to provide the rules to replace some but not all of those in the target p. The rules which can be substituted from q into p are

1. Person A,
2. Person B,
3. Rule of the game to be played (#1 or #2).

Each agent in the game has its own set of these three features. In recombination, while the source agent's features remain unchanged, some (not all) of the target agent's features can be replaced by those from the source agent. In other words, q's Person A (and/or B) may also become p's Person A (and/or B), and/or q's Rule may also become p's Rule. The number of features of the genotype of p which are substituted for those in q is chosen

uniformly randomly (one, two or three features are substituted with probability 1/3 each).

Mutation

If the rules for game i_{n+1} are to be created from a mutation of the rules describing agents in game j_n, then a uniformly random fraction drawn between 0% and 10% of the total number of agents in game i_{n+1} are derived in the following manner. To mutate an agent's genotype, one of three possible mutations can occur with probability 1/3. The mutated feature is either

1. Person *A*,
2. or Person *B*,
3. or Rule of the game to be played (#1 or #2).

If Person *A* or Person *B* is mutated, a random agent different from the agent being mutated is selected to be the new Person *A* or Person *B*. If the Rule is mutated, it becomes either Rule #1 or Rule #2 with probability ½ each.

Results

A number of patterns discovered using the IEC technique described above are presented in this section. Figure 4 shows several such patterns.

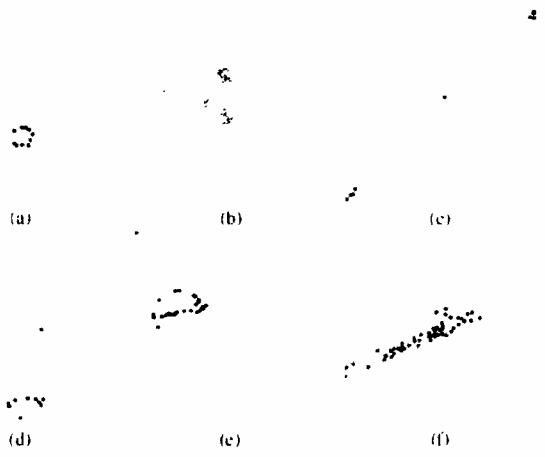

Figure 4: A few examples of evolved behaviors. (a) Circle: agents chase each other around in a circle. (b) Juggle: two blobs fuse and reemerge and sometimes toss a smaller blob at each other. (c) Corner-middle: two groups of agents go to opposite corners while one stays in the middle. (d) Pursuer-evaders: an agent follows a larger group that slows down, is reached by the pursuer, then escapes again. (e) Chinese streamer: a D shape that moves around. (f) Somersault: a thick line that makes a 360 degree turn, then stops, then turns back in the opposite direction.

Figure 5 shows snapshots of the spatio-temporal dynamics of one of the patterns, the Chinese streamer (CS). Figure 6 shows the social network structure (who interacts with whom) of a typical CS pattern. The network has a highly specific structure that makes the CS pattern possible.

Figure 5: "Chinese streamer" pattern. From a random initial placement, a pattern quickly emerges (a-d) and starts turning, stabilizing in a shape with a handle and trailing ribbon which rotates smoothly. The direction of rotation can be clockwise or counterclockwise (as here), presumably depending on the initial positions.

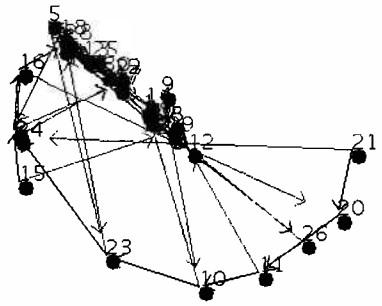

Figure 6: Social network structure of the CS pattern.

The CS pattern is particularly interesting in that it is unexpected (it was impossible to predict that the system could display this type of behavior under the right relationship graph), robust to mutations used to evolve it (a by-product of evolutionary search) and totally insensitive to initial conditions (when the appropriate relationship graph is in place, the Chinese streamer always forms regardless of the initial positions of the participants; initial conditions influence the rotation of the pattern, which can be clockwise or counter-clockwise depending on the details of the participants' locations). It also happens to be robust to noise in the behavior of the participants. The noise-adding procedure that was used is described in Figure 7. We found that below a noise level of ν=2, the Chinese streamer still forms. A noise level of ν=2 is quite large since the random vector (***E***) added to the initial displacement vector (***D***) is up to twice as large as the initial vector.

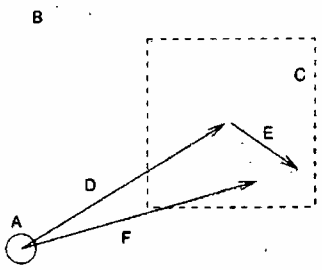

Figure 7: Altering an agent's behavior by adding random noise. Agent *A* is moving at small, discrete displacements of length δ, which is the radius of circle *B*. ***D*** is the intended movement vector as computed by the rules. ***E*** is a random vector uniformly chosen in the square *C* whose side's length is $v\delta$, where v is called the noise level. The vector obtained from adding ***E*** and ***D*** is normalized to obtain ***F***, the final displacement vector.

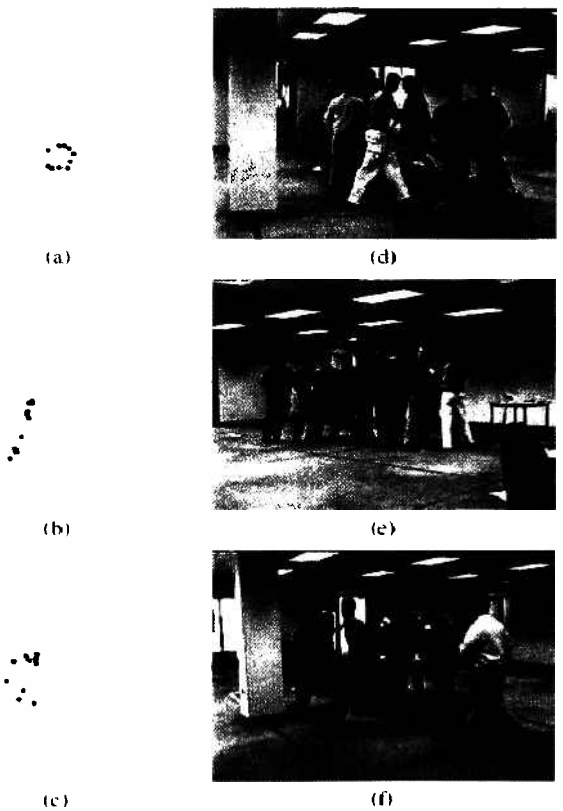

Figure 8: Three rules designed for swarms of 10 agents, evolved using the IEC interface (a-c) and then given to a group of people (d-f). Rule "circle" (a,d) makes all agents run around in a circle; rule "align" (b,e) made them form a straight line and rule "Chinese streamer" resulted in a central cluster with a tail or "ribbon" circling behind.

Lastly, Figure 8 illustrates the ultimate real-world test of the approach: when the rules and interaction graph discovered by applying IEC to the dynamic output of the agent-based simulation are given to humans, the predicted patterns do emerge. Obviously, a reliable agent-based model is necessary to get real-world validation.

5 Discussion

We have illustrated with a simple game example how to tackle the issue of designing decentralized, self-organizing systems with "interesting" properties.

Using this approach requires a shift in mindset from the traditional top-down design approach: here, design is *exploratory* because the aggregate-level capabilities of the system are not known ahead of time. Although it might be possible to create a fitness function *a posteriori* that will make the evolutionary algorithm discover, say, the Chinese streamer automatically (without human intervention), defining such a fitness function requires that the user know the existence of the Chinese streamer as a potential aggregate-level pattern. The exploratory design approach outlined and illustrated in this paper enables the user to navigate the space of possible aggregate-level behaviors without *a priori* knowledge of what is possible or what to look for.

Some of the patterns discovered in the simple game system were extremely surprising and could not be foreseen by any human engineer. Obviously the patterns discovered may sometimes be difficult to understand and need to reverse-engineered. For example the Chinese streamer pattern has been reverse-engineered based on its social network structure (Figure 6) and we now understand its mechanisms.

This approach to designing self-organizing systems has a wide range of applications, from collective robotics to distributed control. One example: radio-frequency tags, known as RFIDs. Although RFIDs have recently become very popular, most users intend to use them with a centralized mindset without knowing what a self-organizing collection of RFIDs might be able to do collectively. Our exploratory design approach enables an open-minded search for the hidden capabilities of such a system.

There are limitations to the approach. First, it requires the complete pre-specification of the micro-rules, that is, one needs to know in advance the complete space of possible microscopic behaviors. Second, it requires a mapping from micro-rules to aggregate-level behavior, such as bottom-up simulation. Third, it requires a way of visualizing the aggregate behavior. Fourth, it can only work with small populations and a small number of generations because of user fatigue. Lastly, it is based on the premise that, although the user may not know ahead of time what aggregate-level behaviors may emerge, she will recognize and interesting pattern when she sees one.

References

Anderson, C. 2003. Linking micro- to macro-level behavior in the aggressor-defender-stalker game. Pages 7-14 in: *Proceedings of the Second International Workshop on the Mathematics and Algorithms of Social Insects* (T. Balch & C. Anderson, eds.), Georgia Tech.

Axelrod, R. 1997. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration.* Princeton University Press, Princeton, NJ.

Bonabeau, E. 2002. Agent-based modeling: methods and techniques for simulating human systems. *Proc. Nat. Acad. Sci. USA* **99**: 7280-7287

Bonabeau, E., Guérin, S., Snyers, D., Kuntz, P., Theraulaz, G. & Cogne, F. 2000. Complex three-dimensional architectures grown by simple agents: an exploration with a genetic algorithm. *BioSystems* **56**: 13-32.

Dawkins. R. 1987. *The Blind Watchmaker.* W. W. Norton, New York.

Epstein J. M., Axtell R. L. 1996. *Growing artificial societies: social science from the bottom up.* MIT Press, Cambridge, MA.

Forrest, S. 1993. Genetic algorithms: Principles of adaptation applied to computation. *Science* **261**: 872-878.

Funes, P., Orme, B. & Bonabeau, E. 2003. Evolving emergent group behaviors for simpe humans agents. Pages 76–89 in: *Proceedings of the 7th European Conference on Arti.cial Life (ECAL 2003)* (P. Dittrich, J.T. Kim, eds.), Dortmund, 14–17 September, 2003.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing.

Helbing, D., Farkas, I. & Vicsek, T. 2000. Simulating dynamical features of escape panic. *Nature* **407**: 487-490.

Palmer, R. G., Arthur, W. B., Holland, J. H., Le Baron, B. & Tayler, P. 1994. Artificial economic life: a simple model of a stockmarket, *Physica D* **75**: 264-274.

Reynolds, C. 1987. Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics* **21**: 25-34.

Sims, K. 1991. Artificial evolution for computer graphics. *Computer Graphics* **25**: 319-328.

Sims, K. 1992. Interactive evolution of dynamical systems. Pages 171-178 in: *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life* (F. J. Varela & P. Bourgine, eds.), MIT Press, Cambridge, MA.

Sims, K. 1993. Interactive evolution of equations for procedural models. *Vis. Comput.* **9**: 446-476.

Still, K. G. 1993. New computer system can predict human behaviour response to building fires. *Fire* **84**: 40-41.

Takagi, H. 2001. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. *Proc. IEEE* **89**: 1275-1296.