Competitive and Cooperative Behavior in Bio-Inspired AI

By Olav Bjørkøy (olavfrih@stud.ntnu.no)
Department of Computer and Information Science
NTNU, January 6, 2011

Abstract: This paper explores the nature of multi-component AI systems inspired by biological solutions to problems also found in computer science. It is shown that by classifying the nature of such components into cooperative and competitive behavior, and by exploring whether this stems from hard-wired properties or emergent behavior, meaningful conclusions on the applicability and expected outcome of these AI systems can be reached. By using the same classifications, advantages and disadvantages of using bio-inspired AI systems for a multitude of problems are discussed.

* * *

What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle. — Marvin Minsky

Techniques in artificial intelligence inspired by biological systems often exploit the task-solving properties of multi-component systems that are common sights in nature.

Nature employs all sorts of systems comprised of multiple components that both work together and compete with each other. In evolution, the components are multiple individuals competing for the chance to pass on their genes. In multicellular organisms, trillions of cells work together to form a fully functioning entity.

The complexity of components in a biological system range from the remarkably simple to the incredibly complex. While creation of complex structures from complex components are interesting, an impressive aspect of nature is how the combination of many relatively simple components perform complex feats of emergent competitive and cooperative behavior.

The creation of multi-component systems where simple components can be used in competition, to find the best component for the task at hand, or in cooperation, where abilities of the components in symphony is greater than the their sum, is needless to say an enticing prospect for problem solving in the field of artificial intelligence.

The next section thoroughly defines the terms *cooperation* and *competition*, when used in the context of multi-component systems. Section 3 presents three examples of artificial multi-component systems based on cooperation, and three based on competition, all inspired by different biological systems. The final section discuss the advantages and disadvantages of cooperative and competitive behavior in bio-inspired AI.

1 Definitions of Behavior

To properly classify interaction between components in bio-inspired AI of having mainly a cooperative or competitive nature, useful definitions of the two terms are needed. The following definitions, paraphrased from the New Oxford American Dictionary [3], will be used throughout the rest of this paper:

- **Competition:** To strive to gain or win something by defeating or establishing superiority over others who are trying to do the same.
- **Cooperation:** To act jointly and work toward the same end.

Two diametrically opposing strategies in behavior is apparent from these definitions: Will components of the system measure their success by comparison with other components, or will performance of the system as a whole be the metric for individual success?

1.1 Competition

In systems driven by competition, each component strives to achieve a goal by defeating or outperforming other components, as measured by a success metric common to all participants. The goal of each component is the same, but the method and rate in which it is achieved compared to competing components is the measure of individual success.

The success metric can be inherent to all components, for example if the goal is to survive as long as possible in an environment of hostile components. In this case, each component knows its objective and is tasked with finding the best or quickest way to achieve this on its own. An external success metric can be used, in which case the system itself test each component in some defined task or for some property. Here, no component is aware of its final purpose, but may be developed and improved through random adjustments or couplings with other components [2, p. 13].

In a system driven by competition, each component thrives on superiority towards its peers. If each component is allowed modifications to improve itself, the system may develop and identify one type of component or configuration best suited to solve the task at hand. This, however, is not the goal of all competition-driven systems. By letting each component improve itself and carry their improvements further into the competition, the performance of every component will improve, resulting in a host of applicable solutions, which may be the ultimate goal. In contrast to a cooperative system, where components may strive for high average performance of the group as a whole, a competitive system rewards components standing out from the pack.

1.2 Cooperation

In systems based on cooperative behavior, components act jointly towards a common end goal. The performance of the system as a whole is paramount, while the performance of each individual is only important in respect to how it helps the entire population achieve the objective. Through cooperation, components can solve problems and exhibit superlinearity [2, p. 532], where the work of the collection as a whole is greater than the sum of work by individual components.

As in systems based on competition, the components themselves can be imbued with the system's ultimate goal, and be tasked with finding the best solution, in this case through cooperation. The goal might require a cooperative solution by itself, but by providing each component methods of communication with other components, emergent cooperative behavior might be what the components decide to be the best solution to the current task.

However, this is not always necessary for complex emergent cooperative behavior to appear. Through simple local rules inherent in each component, the final resulting global behavior might be cooperative in nature. Such behavior is hard to predict, as it is not trivial to foresee what effects local rules will have in a system of multiple agents. However, it can be used to study models where the local rules between components are known but the global dynamics are poorly understood.

2 Examples of Behavior

With proper definitions for competitive and cooperative behavior in multi-component systems, bio-inspired techniques in artificial intelligence can be categorized according to the behavior exhibited by their parts.

2.1 Competitive Behavior

Fitness based selection in genetic algorithms, classification competition in self-organizing maps and forest fire modeling with cellular automata are examples of competitive behavior in multi-component systems.

Fitness Based Selection in Genetic Algorithms

Evolution is a classic example of competition in biology. Since natural resources such as food and land are limited, not all individuals in a population experiencing some shortage of resources will survive long enough to procreate. The individuals most fit in the current environment will survive long enough to pass on their genetic material, which in turn is mixed with the genes from another fit individual, while also being minutely mutated in the process. The result is a new generation of individuals, inheriting features from the most fit individuals from the previous generation, along with some minor variations in the gene pool. Over time, the whole population evolves to better fit their current environment [2, p. 2].

In artificial intelligence, this process is mimicked by genetic algorithms. This type of system is comprised of a population of individuals, which are the components in this system. Each component is a solution to the problem the algorithm is set to solve, for instance the best combination of moves in a game.

Each component carries a genetic string. A global developer creates a fully developed individual, called a phenotype, from the genetic specification. This phenotype is then run through a fitness test, i.e. the problem the algorithm is set to solve, ranking it with respect to the rest of the population. Through a multitude of possible selection mechanisms, the best individuals are selected, their genetic strings combined, and the resulting string, with an optional minute mutation to explore new possible areas of the fitness landscape, is developed into a new phenotype, which is a new solution to the task at hand. See [2, p. 13] for a thorough explanation of genetic algorithms.

This example of bio-inspired AI can be classified as a multi-component system where competitive behavior results in an iterative search of the hypothesis space for the best solution. Each individual's phenotype is a solution, and the many possible solutions in each generation compete through evaluation for inclusion in the next generation. This artificial version of evolution is indeed much simpler than its natural counterpart, but the essential principle remains the same. Through competition, the solutions in the population becomes better over time before hopefully achieving the desired or an acceptable solution to the current task.

If the fitness metric used in competition is external to the population, each individual solution is in practice only interested in how it fares in this test. By always retaining the best solution, the best fitness of the population will be strictly increasing. The fitness metric can also be internal, where it depends not on an external measure, but on how good the rest of the population is in relation to the individual [2, p. 22].

The result of the competition between individuals is not only dependent on the fitness function, but also on the mechanism for selecting the best solutions for advancement to the next generation. There are many such mechanisms, including ranking, roulette-wheel selection and tournaments between subsets of the population [2, p. 23]. These mechanisms define the terms for the competitive behavior.

Competition is explicitly programmed in a genetic algorithm, through fitness testing and the choice of selection mechanism. Competition does not emerge as the chosen method by the individuals to reach the desired goal. However, depending on how the system is programmed, individuals that perform better in the fitness test will be selected, so, for the algorithm to work, it is essential that the fitness test returns an accurate assessment of each individual, to ensure desired competitive behavior.

Classification Competition in Self-Organizing Maps

A self-organizing map (SOM), or Kohonen map, is a type of artificial neural network quite different from conventional networks [2, p. 175]. The purpose of a SOM is to create a discrete representation from input patterns presented during training, which allows the map to classify unknown patterns. A SOM consists of a set of nodes (or neurons) which, unlike in regular ANNs, are linked together forming a topological structure, where each node has a constant set of neighboring nodes. Each node represents a position in a vector space, and by matching new input examples to these vector space coordinates after training, new patterns can be classified [2, p. 206].

SOMs use competitive hebbian weight adjustment during training to move the nodes in the map to positions where they help classify the current training examples. When presented with a new training example, the distance to each node from each element in the training vector is calculated. For each element, the node that wins, i.e. the node that is closest to the element in the training vector, has its weight adjusted so that it moves closer to the element in question. Neighbors of the winning node gets a smaller prize by receiving a small adjustment to their positional weights, nudging them closer to the example they almost won. Note that the topology of the map, represented by the neighborhood of each node, is preserved at all stages during training.

This example of bio-inspired AI can be seen exhibiting competitive behavior during training. For every element in an input vector, each node strives to win by being chosen as the classification for this element. Its reward is a positional adjustment making it more likely to correctly classify similar input elements. By also slightly adjusting weights of neighboring nodes, the competition awards multiple rewards based on how good a match each node provides. Note that in early stages of training, topologically neighboring nodes might not be neighbors in the current vector space, but after a few training examples, this is quickly adjusted so that node proximity in the graph implies proximity in vector space.

Although the rules for updating node positions employ competitive behavior, the final map shows cooperative behavior in the sense that all nodes cooperate to correctly classify the input. This, however, is merely an indication of the underlying hard-wired competition performed during training. As the number of nodes is usually set close to the number of elements in an input vector, every node has a probable outcome of winning the classification for at least one input element. The final cooperative behavior is then an expression of the fact that each node won one class of elements, and can more correctly be seen as hard-wired competitive behavior, where all nodes win something. However, if there are more nodes than vector elements, they can not all be winners.

Forest Fire Modeling with Cellular Automata

Many natural phenomena can be replicated at an abstract level through models of simple components. One such technique is the use of cellular automata (CAs) to model complex emergent behavior from a numerous collection of simple components. In a CA, each component is a cell residing in cellular space, often represented by a 2D matrix [2, p. 102]. At each time step, cell states are updated according to simple built-in rules that dictate cell transformations between a given number of states. The transformation of a cell can depend on properties of its neighbors or a probabilistic transition rule. In the latter case, the system is called a probabilistic CA.

The forest fire CA is one such probabilistic CA. Each cell has a neighborhood of either 4 (von Neumann neighborhood) or 8 other cells (Moore neighborhood). A cell can be in one of three states: Being empty, having a tree, or having a burning tree. At each time step, cells are transformed by the following probabilistic rules [2, p. 126]:

1. A cell with a burning tree becomes empty.
2. A cell with a tree neighboring at least one burning tree becomes a burning tree.
3. A cell with a tree, without any neighboring burning trees, becomes a burning tree with probability $(1 - f)$, where f is the ”probability of lightning”.
4. An empty cell becomes a tree by the ”probability of growth”, g, thus remains empty with the probability $(1 - g)$.

Through these rules, and by tweaking the parameters f and g, the resulting space-time diagram of the cellular space shows patterns resembling the relentless spread of a forest fire, and the following regrowth of trees in the fires path [2, p. 127].

Competitive behavior is apparent in the way fire consume living trees, and in the way tree cells and fire cells fight for survival. In the first case, the trees and fire can be seen as two groups of components struggling for survival. The fire cells aims to consume all trees, while the tree cells wish to populate the entire forrest floor. Through the rules given above, fire needs trees to spread, and trees need empty space to grow. In the second case, the two groups fight amongst themselves for space and survival. A fire cell surrounded by equal cells will die out, as there are no more trees to spread to. Likewise, a tree can only grow in a cell not occupied by another tree.

While this is a crude approximation to the natural world of forest fires, the emergent competitive behavior provides an interesting insight into how such fires spread, and how the forest rebuilds itself. The simple rules only specify how each cell should behave in response to its neighborhood, not the goal or final global behavior. Despite this focus on locality, the resulting emergent competitive behavior shows how fire spreads and trees regrow in a realistic manner.

2.2 Cooperative Behavior

Ant colony optimization, hebbian learning in artificial neural networks and flocking through distributed behavior are examples of cooperative behavior in multi-component systems.

Ant Colony Optimization

Finding the shortest path through a graph with many nodes and connections is a common yet difficult task in computer science, a problem to which nature has found its own solutions. The ant Colony Optimization (ACO) algorithm mimics how ant colonies approach finding the shortest path to a food source.

In nature, ands of a colony initially explore the environment at random, looking for sources of food. Each ant produce pheromones on the ground allowing it to find its way back to the nest. Upon locating food, an ant will carry it back to its nest, following the original path, producing more of the pheromone marking its route. As other ants are conditioned to most likely follow paths with the highest concentration of pheromones, more and more ants will eventually follow the shortest path between food and nest, as this is the path the most ants have already taken. To avoid multiple ants following a suboptimal path to a food source, the pheromone evaporates from the ground after some time. [2, p. 527].

The ACO algorithm looks at an interconnected graph and dispatch virtual ants with virtual pheromones that traverse the graph edges in the same way ants explore territory. Ants return from the target node to the source node, continuously depositing virtual markers, attracting other ants. As in nature, the virtual pheromone is weakened after some time to prevent crowding of a suboptimal path. Before long, ants converge on a single path constructed from the most frequently traversed edges to form the optimal graph path, or at least something quite close to it [2, p. 528].

In this algorithm, each component has no knowledge of the ultimate goal of its existence, only simple, local rules for basic cooperative behavior to exhibit in certain conditions. Initially the graph is searched at random. When the target is found by a single component, its steps are retraced, making that particular path more attractive in the eyes of other components. When multiple components locate the target through different paths, pieces of each path can be stitched together forming an even shorter alternative. This is achieved in totality through the rules for virtual pheromones and the behavior that ants perceive paths with more pheromone are probably better than unexplored paths.

This algorithm shows emergent cooperative behavior. Components (ants) in the system have no direct knowledge of what they achieve by their behavior, only that following in the steps of many other components is a good idea. While the rules that specify how virtual ants should cooperate by sticking together could be seen as hard-wiring, and not emergent behavior, the hard-wired rules are indeed so simple that the impressive level of cooperation exhibited by the entire colony is hardly described by it in full.

The emergent intelligence shown by a cooperative effort to find the shortest route in the graph mimics that of ACOs biological counterpart quite closely. By working together, the components quickly find a solution to a difficult global problem through the use of simple local rules, and without any form of global control or instruction.

Hebbian Learning in Artificial Neural Networks

Biological neural networks are extremely complex structures science is far from understanding completely. Hebbian learning in artificial neural networks can be seen as a coarse approximation to how learning is performed in, among other areas, the visual and auditory cortex of the mammalian brain when presented with new stimuli [2, p. 198]. Hebb's rule states:

> *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells, such that A's efficiency as one of the cells firing B, is increased.*

Another formulation says that neurons that "fire together, wire together". Although only one of many theories on biological learning, the changes in neurological pathways predicted by this rule seem to be an important part of how mammals learn to associate concepts such as words, images and object features by repeated exposure. When neurons partake in the same association process with current stimuli, their connection is strengthened by a metabolic change in the neurons and their synapses.

In artificial neural networks, hebbian learning is one technique for unsupervised learning. Patterns in input vectors, such as the association for mapping an input image to the correct action, can be learned through hebbian weight adjustments. The adjustment comes in many variations, but in its simplest form, the adjustment is

$$\Delta w_i = \lambda u_i v \tag{1}$$

where Δw_i is the weight change, λ is the learning rate, u_i is the output of the presynaptic neuron and v is the output of the postsynaptic neuron [1, p. 12].

The ANN is presented with repeated input examples representing one pattern the elements of which it should learn to associate with each other. After repeated exposure, the weights in the neural net will have adjusted to a state where this pattern will be presented when the network is given a new example of the pattern, even if this pattern is corrupted by noise. The pattern now exists as knowledge within the network.

In this example, components of the system, the neurons, cooperate to achieve their common goal through simple local learning rules. Initially, neuron A cooperates with neuron B since it firing means the efficiency of B is increased, as stated in Hebb's rule. Secondly, the neurons cooperate by using both output values to adjust their common weight as by Equation 1, ensuring that the same firing will be more efficient during future input.

The cooperative nature of hebbian learning is hard-coded into how weights on arcs between nodes changes during training. Nodes do not decide how weights should be changed on their own, but follows a predetermined equation for learning the current patterns. If coupled with a genetic algorithm, this equation can change as part of the many solutions represented by individuals in the population, but the finished neural net follows the rules provided at generation time.

Flocking Through Distributed Behavior

Another example of cooperative behavior in multi-component AI systems is the distributed behavioral model of coordinated exploration in swarm robotics [2, p. 531]. An example of this are the "boid" components, created by Craig W. Reynolds. A boid is a simple component in a multi-agent system, which can model behavior in, among other things, flocks of birds, schools of fish and groups of particles.

Each boid follows a simple set of local behavioral rules which result in a flock-like behavior on a global scale. The seemingly complex movement achieved by the flock as a whole can be attributed to the simple behaviors innate in each simulated boid. At the lowest level, each boid adheres to three rules of operation [4]:

Separation: If a boid comes too close to another boid, it will steer away to avoid collision.

Alignment: Each boid will follow the average heading of its nearest neighbors in the flock.

Cohesion: If a boid finds itself too far from other boids, it will steer towards the average heading of the boids in its neighborhood.

What each boid "sees" is determined by how many other boids it considers to be in its neighborhood. This parameter is decided by a distance value, which represents how far the boid can see, and an angle, describing the width of its field of view. The boids can also have mode complex rules, such as predictive obstacle avoidance and locating a common goal. A video of such behavior can be seen at [5]. The boids appear to follow the same rules of flocking as observed in nature, where the main group sticks together, follows the same heading despite fairly rapid turns. Boids that find themselves outside the main flock quickly return to the others.

The neighborhood definition combined with each boid's simple rules for behavior results in cooperative behavior. The flock twists and turns through space in a tight, often seemingly planned formation. The intelligent flocking exhibited by the population of boids stem from simple local rules that only describe their behavior towards their small number of neighbors. This is then a good example of emergent cooperative behavior. The flock can act towards a common end in an intelligent way, be it obstacle avoidance, locating a goal or simply sticking together.

3 Advantages and Disadvantages

There are many advantages and disadvantages of using cooperative and competitive dynamics in bio-inspired AI, some general and some based on specific use of the system in question. Considering the hard-wired or emergent nature of the component behavior is also important when evaluating metrics such as run-time, efficiency and development time.

3.1 Advantages

The main advantage of a multi-component behavioral AI system is the ability to model complex global behavior, and locate hard to find solutions to difficult problems, through the use of relatively simple parts. The nature of each system's components range from the simple, as in the ant colony optimization algorithm, to the relatively complex competitive mechanics of genetic algorithms, as discussed in Section 2. However, while each component may be complex, the nature of the problems they can solve often rank in the top tier of difficult problems in computer science.

In AI systems based on evolution, with relatively complex components, the main advantage is the ability to locate solutions to problems through iterative improvements and by searching in an often infinite search space. The designer of the system does not necessarily need to know how a good solution can be found, only how to represent the problem in terms of genotypes, developers and phenotypes. The inherent nature of evolution takes care of the rest. In this case, hard-wired methods of selection to some degree ensure that a good solution will be found, and the same mechanisms can be tweaked to ensure proper exploration of the fitness landscape. With systems sporting emergent behavior, performing such searches and guaranteeing a good solution is much more difficult.

In AI systems sporting less complex components, like those found in cellular automata and swarm robotics, the components achieve accurate modeling of complex global behavior through simple local rules. Programming such systems from the bottom up makes modeling tasks achievable, where if the system was engineered from the top down, the problem might be infeasible due to its complexity. Through emergent cooperative or competitive behavior, natural phenomena can be modeled before their global dynamics are fully understood. Contrast this to hard-wired approaches, where the developer must have a good understanding of what is to be modeled to properly configure the wiring.

3.2 Disadvantages

While the advantages of multi-component systems may be plentiful, important disadvantages often prohibit the use of these algorithms and dynamics in production systems. The most important drawbacks concern run-time, no guarantees of optimal solutions, behavior that is difficult to predict and system-design time.

In AI systems based on evolution, the lack of guarantees concerning the optimality of the found solution is an important problem. While such systems can be set to explore large areas of the hypothesis space (i.e. the fitness landscape), there is no guarantee that the best, or even a good solution, will be found. The dangers of getting stuck at local maxima is always present, often requiring multiple runs of the same algorithm, since evaluating positions in the fitness landscape is no easy task. For this reason, genetic algorithms often remain tools for exploration of design choices and parameter approximation, as opposed to being used in real-time systems. The selection and exploration mechanisms may be hard-coded, but there is no way to ensure the returned solution will be the one desired, without incurring significant drawbacks in run-time.

In AI systems based on less complex components, like the forest fire CA or in multi-layer neural networks, the main disadvantage is often the difficulty in predicting the global behavior of the population based on simple local transition rules. Since the main purpose of many such systems is displaying emergent cooperative or competitive behavior to model a phenomenon that is difficult to engineer top down, designing the components to achieve some desired global behavior is by definition difficult. This is not helped by the fact that, at least in the case of CAs, the resulting systems are far from deterministic – one run of the algorithm will not ensure that it is working properly. Due to difficulty of reverse-engineering desired emergent behavior, such algorithms is mainly used for studying how local rules affect global behavior.

The same problem persists in artificial neural networks. There are no rules for how many internal nodes or hidden layers achieve the best run-time and solution optimality for a given problem – a question often solved through trial and error. By training the networks through hebbian learning and backpropagation, or by evolution, the layer, node, connection count can be dynamically decided. However, this kind of machine learning still does not solve the complex problem of getting stuck at local minima or maxima.

In many examples of multi-component systems, run-time is also an important and problematic factor. The iterative behavior and uncertain amount of time before a desired state or solution is reached, makes predicting the actual runtime of the system difficult. One solution is to limit the number of generations or epochs, but this may have a further decremental effect on the validity of the found solution. In hard-wired systems, run-time can more easily be controlled, as it remains in the hand of the developer. With emergent behavior, run-time is mainly dependent on how long it takes for the desired behavior to appear os stabilize, a timeframe that might be different between iterations.

4 Conclusion

Competitive and cooperative behavior in multi-component systems is a great way to solve many important problems in different fields of computer science. Novel approaches to exploring a hypothesis space or modeling global dynamics between simple agents gives system designers new methods in solving problems that can be seen in an entirely different light.

Knowing the advantages and disadvantages of multi-component bio-inspired AI is essential before venturing out into this vast field of new and exiting techniques. In some tasks, like modeling complex uncharted global behavior through emergent behavior, or finding solutions to problems that conventional algorithms have a hard time tackling, these systems achieve as good or better than many established approaches. On the other hand, the disadvantages with respect to predicting emergent behavior, ensuring an optimal solution and calculating expected run-time, must be taken into account.

References

[1] Keith Downing. Introduction to artificial neural networks. 2009. Available from `http://www.idi.ntnu.no/emner/it3708/lectures/ann-intro.pdf`, accessed 22.04.2010.

[2] Dario Floreano and Claudio Mattiussi. *Bio-Inspired Artificial Intelligence*. The MIT Press, 2008.

[3] Erin McKean, editor. *The New Oxford American Dictionary, Second Edition*. Oxford University Press, 2005. Accessed via Mac OS X's Dictionary application.

[4] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. 1987. Available from `http://www.red3d.com/cwr/papers/1987/SIGGRAPH87.pdf`, accessed 22.04.2010.

[5] Craig W. Reynolds. Simulated boid flock avoiding cylindrical obstacles, 1987. Available from `http://www.siggraph.org/education/materials/HyperGraph/animation/art_life/video/3cr.mov`, accessed 22.04.2010.