Computing Networks: A General Framework to Contrast Neural and Swarm Architectures

Carlos Gershenson[1,2]
[1] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
Ciudad Universitaria
Apdo. Postal 20-726 / Admón. No. 20
01000 México D.F. México
cgg@unam.mx `http://turing.iimas.unam.mx/~cgg`
[2] Centro de Ciencias de la Complejidad
Universidad Nacional Autónoma de México

January 28, 2010

Abstract

Computing Networks (CNs) are defined. These are used to generalize neural and swarm architectures, namely artificial neural networks, ant colony optimization, and particle swarm optimization. The description of these architectures as CNs allows their comparison, distinguishing which properties enable them to perform complex computations and exhibit complex cognitive abilities. In this context, the most relevant characteristics of CNs are the existence multiple dynamical and functional scales.

1 Introduction

The complex behavior exhibited by swarms has been actively studied in recent decades (Hölldobler and Wilson, 1990; Aron et al., 1990; Reznikova, 2007; Ryabko and Reznikova, 2009) and exploited in engineering (Bonabeau et al., 1999; Dorigo and Stützle, 2004). Recent research has highlighted the similarities between swarms and brains (Couzin, 2009; Marshall et al., 2009; Passino et al., 2008; Trianni and Tuci, 2009). Contributing to the effort of understanding these similarities, with biological and engineering aims, this paper generalizes models of swarm and neural architectures. In particular, artificial neural networks (ANNs), ant colony optimization (ACO), and particle swarm optimization (PSO) are described under the same general framework. The generalization, named *computing networks* (CNs), provides a common ground for comparison and for studying the underlying mechanisms and computational properties common to swarms and brains.

In the next section, the concept of computing network (CN) is defined. In the following sections, this concept is used to describe ANNs, ACO & PSO. Then, there is an extended comparison and discussion. In this section, similarities and differences of the architectures are explored, followed by the discussion multiple dynamical and functional scales. Then, the suitability and equivalence of different architectures is considered. The discussion continues with the cognition of swarm and neural architectures and finishes with an examination of alternate descriptions of the architectures. Conclusions close the paper.

2 Computing Networks: A General Descriptive Framework

Many systems can be described as networks, i.e. nodes connected by edges (Newman, 2003; Newman et al., 2006). In this paper, we use the concept of *computing network* (CN) as a generalization of artificial neural networks (Rumelhart et al., 1986; Hopfield, 1988), ant colony optimization (Dorigo et al., 1991; Dorigo and Stützle, 2004; Dorigo and Blum, 2005; Dorigo, 2007), and particle swarm optimization (Kennedy and Eberhart, 1995, 2001; Dorigo et al., 2008). In this way, the similarities and differences between these characteristic models of neural and swarm intelligence are studied under the same formalism.

A computing network $C(N, K, a, f)$ is defined as **a set of nodes N linked by a set of edges K used by an algorithm a to compute a function f. Nodes and edges can have internal variables that determine their state, and functions that determine how their state changes**. This is a very general definition, and can be applied to describe many architectures and models beyond those discussed in this paper. Computing networks can be stochastic or deterministic (depending on the determinism of functions and algorithms), synchronous or asynchronous (depending on the updating used for the change of states of nodes and edges (Gershenson, 2002, 2004b)), discrete (Wuensche, 1998) or continuous (depending on the type of variables of nodes and edges).

3 Artificial Neural Networks

Artificial Neural Networks (ANNs) were originally proposed as logical models of the neocortex (McCulloch and Pitts, 1943). However, their computing power (Hopfield, 1982) has shifted the research focus from their plausibility as neural models to their application in different fields. There are many different types of ANNs, with different properties and implementations. Here there will be no focus on any particular type of ANN.

In an ANN instantiation of a CN, *nodes* are neurons or units. Each neuron i typically has a continuous state (output) determined by a function y_i which is composed by two other functions: the weighted sum S_i of its inputs $\bar{x}_i$ and an activation function A_i such as the hyperbolic tangent. Directed *edges* ij (synapses) relate outputs y_i of neurons i to inputs x_j of other neurons j, as well as external inputs and outputs with the network. Edges have a continuous state w_{ij} (weight) that relates the states of neurons. The *function* f may be given by the states of a subset of N (outputs $\bar{y}$), or by the complete set N. ANNs usually have two dynamical scales: a "fast" scale where the network function f is calculated by the functional composition of the function y_i of each neuron i, and a "slow" scale where an *algorithm* a adjusts the weights w_{ij} (states) of edges. There is a broad diversity of algorithms a used to update weights in different types of ANN. Figure 1 illustrates ANNs as CNs.

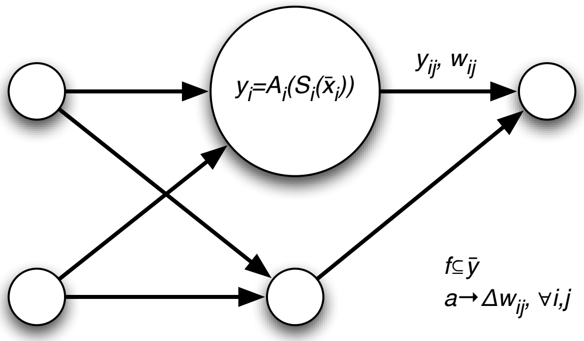

$y_i = A_i(S_i(\bar{x}_i))$
y_{ij}, w_{ij}
$f \subseteq \bar{y}$
$a \rightarrow \Delta w_{ij}, \forall i,j$

Figure 1: Schematic of an ANN instantiation of a CN. Nodes have a function y_i that is computed from its inputs $(\bar{x}_i)$. Edges have weights w_{ij} to determine the importance of the interaction and also carry the output of neurons and network inputs. The network function f or output is given by a subset of node functions $\bar{y}$. The algorithm a changes weights on edges.

4 Ant Colony Optimization

Ant colony optimization (ACO) is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems (Dorigo, 2007). ACO is inspired in the collective behavior of ants and their stigmergic interactions through pheromones.

In an ACO instantiation of a CN, *nodes* are locations that contain a list of "artificial ants" at their location. Each ant k has a path which represents a partial solution s_k^p, from which variables such as distance travelled and nodes visited can be extracted. *Edges* (trails) have two variables: heuristic value η_{ij} (e.g. distance or cost between two nodes) and pheromone value τ_{ij}. There have been different *algorithms* proposed to calculate *function* f, which is given by the shortest path found. In ACO there are also two timescales: a "fast" one in which ants travel through the network, generating paths (solutions) by choosing edges probabilistically at each visited node depending on their state η_{ij},τ_{ij}, and a "slow" one, where the pheromone values τ_{ij} of edges are updated. This is similar to weight adjustment in ANNs. The pheromone update consists of an "evaporation" phase, where all levels are reduced (similar to "forgetting" in some ANNs) and an additive phase (similar to "reinforcement" in some ANNs), where pheromone levels associated with good solutions are increased. In some versions of ACO, there is a "middle" scale, where "demon" (problem specific) actions are taken, such as the application of a local search (Dorigo, 2007). Figure 2 illustrates ACO as a CN.

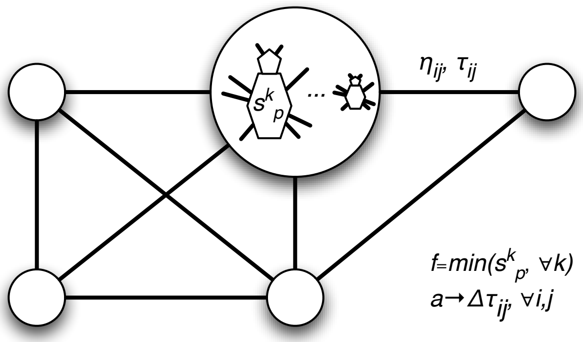

s^k_p
...
η_{ij}, τ_{ij}
$f=min(s^k_p, \forall k)$
$a \rightarrow \Delta\tau_{ij}, \forall i,j$

Figure 2: Schematic of an ACO instantiation of a CN. Nodes contain ants that construct paths s^p_k. Edges contain heuristic η_{ij} and pheromone τ_{ij} values. The function f is given by the best path found. Algorithm a adjusts pheromone concentrations τ_{ij}.

5 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based stochastic approach for solving continuous and discrete optimization problems (Dorigo et al., 2008). It was originally inspired by flocking algorithms (Reynolds, 1987) and social psychology research. In PSO, “particles” move in a search space. Their position represents a candidate solution. Particles adjust their position and velocity depending on their neighboring particles.

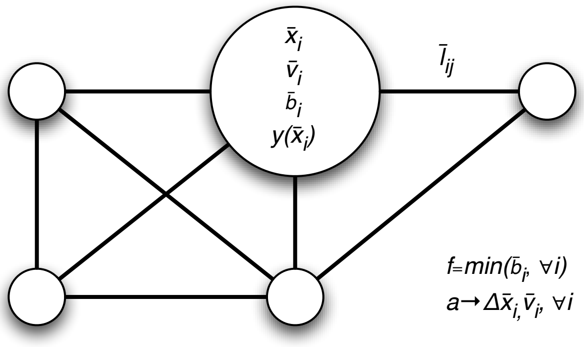
In a PSO instantiation of a CN, *nodes* are particles with position $\bar{x}_i$, velocity $\bar{v}_i$, value of the best solution found $\bar{b}_i$, and a function $y(\bar{x}_i)$ that the network is trying to optimize. The position $\bar{x}_i$ represents a tentative solution. The function f is simply the best solution found by N. *Edges* represent the relationships between neighboring particles. Typically they contain information about the neighborhood’s best solution, which can be represented as $\bar{l}_{ij} = max(\bar{b}_i, \bar{b}_j)$ for nodes i, j related by edge ij. There is a variety of *algorithms* to relate the way in which particles adjust their state. Again, two timescales can be identified: a “fast” one, where particles evaluate the function they are trying to optimize $(y(\bar{x}_i))$, and a “slow” one, where the velocity and position of particles are adjusted by algorithm a depending on their previous states and those of their neighbors (links). Figure 3 illustrates PSO as a CN.

For PSO, hypernetworks (Johnson, 2010) can be used as generalization, so that a single edge can link more than two nodes and to represent the best solution of a neighborhood $\bar{l}_j$.

6 Comparison and Discussion

Table 1 shows a comparison of the language used to relate ANNs, ACO, and PSO in terms of CNs. It can be seen that all three architectures have the same basic components: nodes, edges, an algorithm, and a function. However, there are differences in the particularities of each architecture.

ACO and PSO have been used mainly for optimization. This explains why their f is the minimum (best) of the solutions found. In contrast, ANNs have been used to solve



$\bar{x}_i$
$\bar{v}_i$
$\bar{b}_i$
$y(\bar{x}_i)$
$\bar{l}_{ij}$
f=$min(\bar{b}_i, \forall i)$
$a \rightarrow \Delta\bar{x}_i, \bar{v}_i, \forall i$

Figure 3: Schematic of an PSO instantiation of a CN. Each node contains the position $\bar{x}_i$ and velocity $\bar{v}_i$ of a particle, as well as its best solution found $\bar{b}_i$ and a function $y(\bar{x}_i)$. Edges contain the neighborhood's best solution $\bar{l}_{ij}$. Function f is the best solution found by all particles. Algorithm a changes the position $\bar{x}_i$ and velocity $\bar{v}_i$ of particles depending on the values of their neighbors.

Table 1: Particular instantiations of CNs: ANN, ACO, and PSO.

CN	**ANN**	**ACO**	**PSO**
Nodes	Neurons or units (function $y_i = A_i(S_i(\bar{x}_i)))$	Nodes (ants k (path s_k^p))	Particles (position $\bar{x}_i$, velocity $\bar{v}_i$, best solution $\bar{b}_i$, function $y(\bar{x}_i)$)
Edges	Synapses (weight w_{ij}.)	Trails (heuristic value η_{ij}, pheromone concentration τ_{ij}.)	Relationships (neighborhood's best solution $\bar{l}_{ij}$)
Algorithm	Adjust edges (Δw_{ij}.)	Adjust edges ($\Delta \tau_{ij}$.)	Adjust nodes ($\Delta\bar{x}_i$, $\Delta\bar{v}_i$)
Function	Composition of functions of nodes	Shortest path ($min(s_k^p)$)	Best solution ($min(\bar{b}_i)$)

many different tasks, e.g. classification, generalization, recognition, error correction, and time sequence retention. Still, all the architectures can be described as computing a function f in a distributed fashion. This is because they require the interaction of nodes to produce f.

It is interesting to note that, even when ACO and PSO are inspired by swarming systems, algorithms of ANN and ACO are more similar between themselves than with PSO, in the sense that they update edges, while PSO algorithms update nodes. However, if we decided to extend the models from networks to hypernetworks (Johnson, 2010), where there is a duality between nodes and edges, i.e. one can exchange nodes and edges while preserving the functionality of the hypernetwork. In this case, then PSO particles can be described as hyperedges, and their interactions as nodes. Then, the PSO algorithm a would update edges.

6.1 Dynamical scales

One common characteristic among all three architectures studied is that they have "slow" (a) and "fast" (f) dynamical scales. This is no coincidence. Having multiple dynamical scales is a requirement for computing complex functions. If there is only change at a single scale, then the phase space of f can be explored, but it cannot be changed. Having two dynamical scales, one can explore changes in the phase space of f as well. This property is essential when f is not known beforehand: the algorithm explores different phase spaces until one that satisfies f is found.

The tasks solved by real neural and swarm systems also need to exploit the advantages of multiple dynamical scales. In the case of neural systems, learning (synapse modification) enables the correct adjustment of a particular function of a circuit, e.g. categorization. For swarming insects, local interactions (direct or stigmergic) enable the colony to make complex decisions, e.g. choosing a new nest.

Would it be useful to have three dynamical scales? This would imply the exploration of changes in the space of phase spaces of f. For example, this is used in "evo-devo" (Fontana, 2002; Munteanu and Solé, 2008) or epigenetic (Balkenius et al., 2001) algorithms, where there is a function f, its phase space is explored through the "lifetime" of an "organism" (learning), and the space of possible organisms is explored at an evolutionary scale, e.g. with evolutionary algorithms. Figure 4 illustrates the change possible at one, two and three dynamical scales.

A question that arises is whether CNs with three dynamical scales are computationally equivalent, or more powerful, than CNs with two dynamical scales. The reader is invited to ponder on this question, which is already out of the scope of this paper.

6.2 Functional scales and the relevance of interactions

Apart from having multiple dynamical scales, CNs have multiple functional scales. The most clear scales are those of node (local) and network (global). Subnetworks, modules, layers, or motifs can also form intermediate scales. In CNs, nodes compute certain "local" functions. These functions are combined to produce the CN's "global" function f. However, f cannot be *reduced* to the node functions alone. Since the states of the nodes depend on other nodes, *interactions* are relevant to determine the future state of nodes, and thus f.

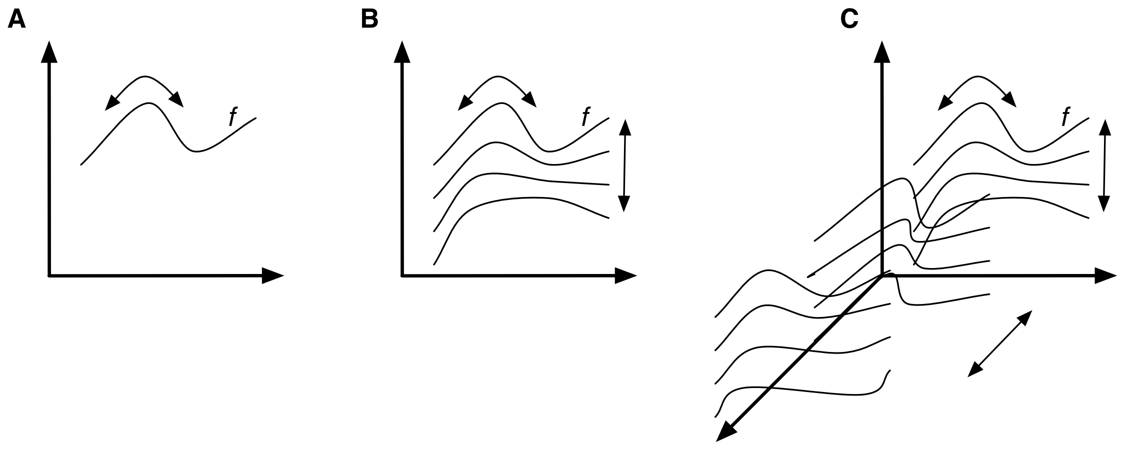


Figure 4: Changes at different dynamical scales: (A) single scale: values can vary only along f, (B) double scale: apart from changes along f, f can also be varied, and (C) triple scale: changes in ways in which f can be varied can also be explored. Note that these diagrams are only illustrative. f can certainly be multidimensional, i.e. in $\mathbb{R}^n$.

As in the case of dynamical scales, having multiple functional scales is a requirement for computing complex functions. In this context, interactions can be described as operators. Local structures (e.g. nodes, motifs) can store certain information and can compute certain functions. However, in many cases, the information produced by local structures is less complex than the one that produced by the global structure (i.e. network). This is because the interactions between local structures integrate information produced at the lower scales to compute the global f. The exceptions are trivial, e.g. when all the interactions are weak or absent, or the local structures are redundant. In these cases, one can say that the complexity of the local structures is the same as the complexity of the global one.

This will be clearer introducing a definition of what is meant by complexity: *Complexity is the amount of information necessary to describe a phenomenon at a particular scale* (Bar-Yam, 2004; Prokopenko et al., 2009; Gershenson, 2007b). With a CN, in most cases more information is necessary to describe the whole network than the collection of all its nodes, namely because of the information contained in edges, which represent interactions. Repeating what was stated above, f cannot be reduced to N only, namely because of K.

A clear example of the relevance of interactions can be seen with cellular automata (CA) (von Neumann, 1966; Wolfram, 1986; Wuensche and Lesser, 1992; Wolfram, 2002), which can also be described in terms of CNs. The states of cells (nodes) depend on the state of their neighbors (edges) according to a certain rule. In the case of elementary cellular automata (ECA) 110 (Wolfram, 2002; Juárez Martínez et al., 2007), the state at time $t + 1$ of each cell depends on its state and of its closest neighbors (3 cells in total) at time t. The updating is done synchronously according to the values shown in Table 2. Figure 5 shows the temporal evolution of ECA 110 for a particular initial state. Even when the behavior of ECA 110 is determined by very simple rules, it is capable of universal computation (Cook, 2004), exploiting the interactions between emergent structures (Juárez Martínez et al., 2007) that arise from the simple interactions of the local neighborhoods.

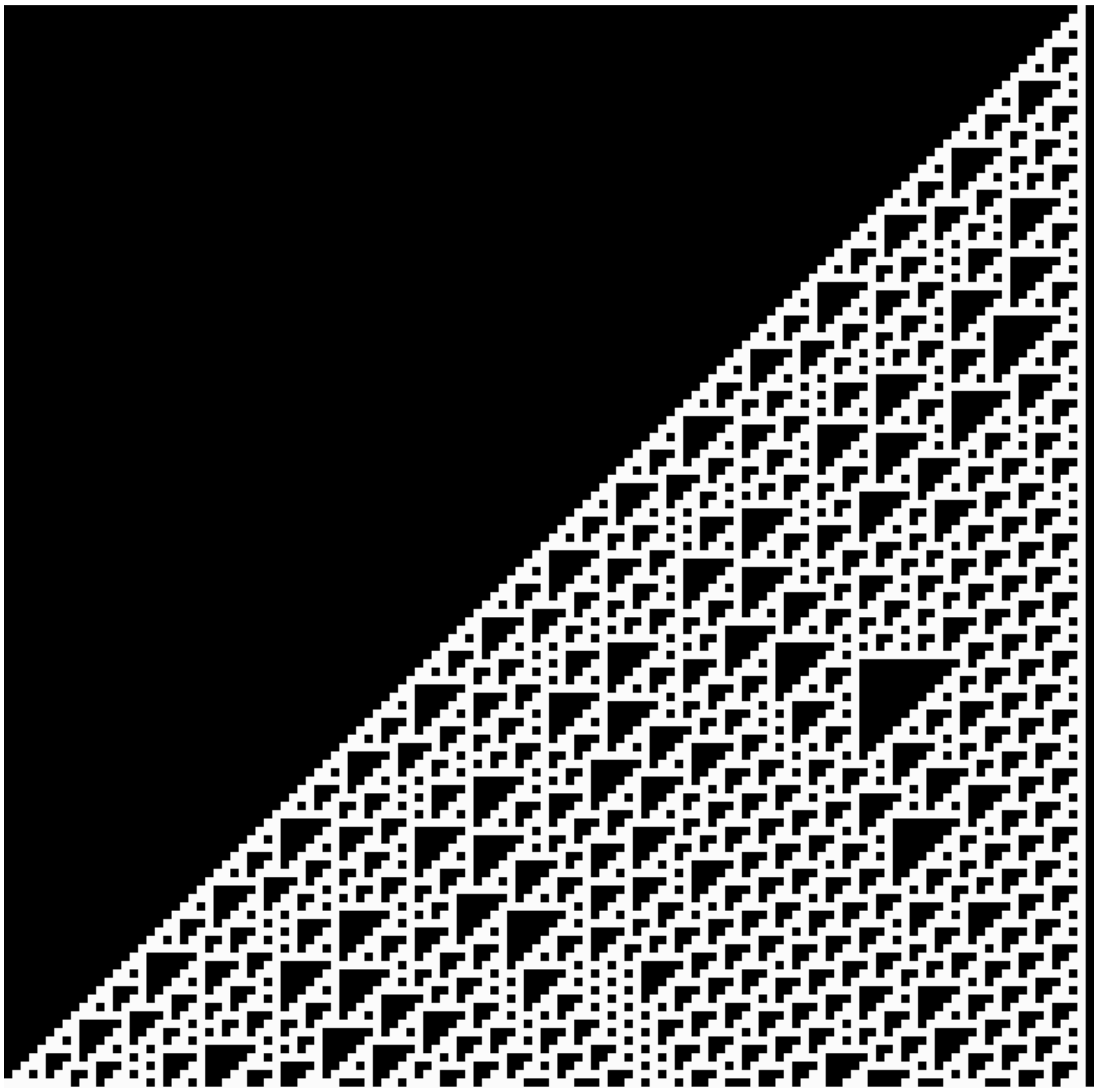


Figure 5: Temporal evolution of ECA 110. Each cell is represented by a column and time flows downwards, i.e. each row represents the state of the CA at successive time steps. Black cells represent '0' and white cells represent '1'. The first row (initial state) consists of a single '1'. The state of other rows depends on the state of the row above. It is not possible to compute *a priori* the state of the last row from the first row without computing all the intermediate states.

Table 2: ECA 110 lookup table. The first column shows the eight possible states of the 3 cells used to update every cell, while the second column shows the state of the updated cell.

t	$t+1$
000	0
001	1
010	1
011	1
100	0
101	1
110	1
111	0

With ECA 110, the relevance of interactions is clearly seen. CNs with simple nodes and functions are capable of complex computations because of the relevant information contained in edges. Note that interactions are not necessarily physical, but they are real. For different systems, there are different "implementations" of edges, e.g. synapses, pheromones, or cues (Couzin, 2009). Still, they all have the same role: to relate states of nodes to compute a distributed function f. With the CN formalism, it is clear that the computational power of a brain is much more complex than that of a large collection of isolated neurons, and the computational power of a swarm is much more complex than that of a group of isolated insects.

For functional scales, we can also ask whether only two scales are less powerful than more than two scales. However, again, the question is beyond the scope of this paper.

6.3 Which architecture is the best?

One might wonder which architecture—ANNs, ACO, or PSO—is the best. There is no best architecture independently of a specific context (Wolpert and Macready, 1995, 1997; Gershenson, 2004a). Different implementations of CNs will be more adequate for different problems, either giving better solutions, or improved speed. The convenience of a particular architecture does not depend only on the problem: different methods will be more useful for different people, depending on their experience and expertise.

A valid question would be: which architecture—ANNs, ACO, or PSO—is more computationally powerful? Since the architectures are so general, it can be conjectured that they all are capable of (theoretical) universal computation (Turing, 1936). There are several ways to show this: each architecture can implement a Turing machine, calculate any computable function, or implement ECA 110, which is already capable of universal computation (Cook, 2004). Moreover, one could implement e.g. an ANN based on ACO or PSO, e.g. where the function of a node is itself determined by an ACO or PSO CN. Similarly, one can implement an ACO or PSO based on ANNs. Finally, one can also develop ACO based on PSO and vice versa. It might not be useful at all, but the idea shows that computationally (in Turing's sense) they all have similar capacities. There will be more differences on particular implementations of ANNs (e.g. given by number of nodes and edges) than between a given ANN and an equivalent ACO or PSO.

The literature is rich in examples of hybrid systems, where some properties of one architecture are combined with those of another one, e.g. (Kennedy and Eberhart, 1995; Wang et al., 2004; Chen et al., 2004; Blum and Socha, 2005; Mozafari et al., 2006) to cite a few of them. Actually, the original PSO paper (Kennedy and Eberhart, 1995) used PSO as an example to train an ANN. This illustrates that for a particular problem and for a particular expertise of the developers, no single approach gives the best solutions.

Having discussed the similarity of the computational capacities of neural and swarm architectures, we can continue with the discussion about the role of the architectures in cognition.

6.4 Cognition

Cognition comes from the Latin *cognoscere*, which means "get to know". We can say that **a system is cognitive if it knows something** (Gershenson, 2004a). With this definition, it is not possible to draw a boundary between cognitive and non-cognitive systems. Since somebody has to judge whether a system knows or not, it is partly observer-dependent. Instead of discussing whether a system is cognitive or not, it is more fruitful to distinguish different types of cognition (e.g. human, animal, biological (including plant and bacterial), social, artificial, adaptive, systemic (Gershenson, 2004a)), to compare and better understand them.

From this perspective, it is clear that swarms are cognitive systems because they *know* how to forage, find sites, build nests, and even add and subtract small numbers (Reznikova, 2007; Ryabko and Reznikova, 2009). Neural architectures are cognitive because they *know* how to categorize, classify, remember, etc. (Hopfield, 1982). To compare both types of cognition, we can use the concept of computing networks proposed in this paper.

Cognition can be seen as the ability to compute a function f. This is because if a system can compute f, we can say that it *knows* how to calculate f. This vocabulary does not aim at ascribing to CNs a "mind", "consciousness", or other difficult-to-define property usually associated with human cognition. The aim of this use of language is to be able to compare the cognitive capacities of neural and swarm architectures. As discussed in the previous subsection, neural and swarm architectures have similar computational abilities, shown by their generalization as CNs. If we describe cognition as computation, it naturally follows that neural and swarm architectures have similar cognitive capacities, *in theory*. In practice, different implementations will have different cognitive abilities, just as a human brain has different abilities as a rat brain: the former is potentially better at poetry, the latter is potentially better at navigation.

The great advantage of swarm and neural cognition is that they manage to exploit the benefits of multiple functional and dynamical scales to exhibit complex cognitive abilities. As discussed above, multiple scales enable CNs to compute more complex functions. In cognitive terms, this enables neural and swarm architectures to exhibit a more complex cognition, as compared to a system with a single functional or dynamical scale, e.g. a thermostat. We can see that there are cognitive systems with more than two scales, e.g. group cognition (Stahl, 2006), which exploit and combine the cognitive abilities of a collection of humans.

Naturally, swarms are another example of multiple scale cognition, since the cognition of individual insects is provided by a neural architecture.

6.5 Alternative descriptions

The description of ANN, ACO, and PSO in terms of computing networks is only one of several possible languages that can be used to compare the architectures. For example, a *multi-agent* description can be also used: Nodes can be described as agents and edges can be described as interactions. An algorithm regulates the interactions between agents to reach a global state (equivalent to function f). This global state can be described as being reached by self-organization (Gershenson and Heylighen, 2003). This self-organization in a multi-agent system is comparable to the distributed computation of f. The system can compute the same function f, only the description changes. For the purposes pursued in this paper, the network description seems more appropriate. A multi-agent description can be valuable in the process of designing algorithms, since goals of agents and systems can be defined. Then, the algorithm should minimize "friction" (i.e. negative interactions) and promote "synergy" (positive interactions) (Gershenson, 2007a). This will necessarily increase the system's "satisfaction", which is basically what we want the system to do, i.e. f.

Yet another description that can be used is that of *information* (Gershenson, 2007b). Nodes, edges, algorithms, and functions can be all seen as information, while computation is simply a change of information. This is a more general description, so it is not so useful for making a comparison as the one presented here. The information framework might be useful for finding general principles across disciplines, since everything can be described in terms of information.

Different descriptions are suitable for different contexts (Gershenson, 2004a), and the one of computing networks was developed specifically to compare neural and swarm architectures. It will not be as good as their original descriptions for developing e.g. new learning algorithms in ANNs or new optimization algorithms in ACO. This is because the computing networks description is more general and vague than an actual instantiation of an ANN or PSO. More details are required at the implementation level, which were neglected here. The goal of defining CNs is more theoretical than practical: to understand the similarities and differences of neural and swarm architectures, not to improve current technical algorithms.

7 Conclusions

As Trianni and Tuci suggest (Trianni and Tuci, 2009), the principles of swarms can be useful tools for studying the neuroscientific basis of cognition. Here it was shown that both swarm and neural architectures share similar computational and cognitive abilities. This was achieved by defining computing networks (CNs), which are able to generalize neural and swarm architectures, allowing their comparison. By studying the general principles that enable CNs to perform complex computations, one can understand better what are the requirements of neural and swarm systems to exhibit complex cognition. In this paper, we discussed the importance of having multiple dynamical and functional scales to achieve this. From a cognitive perspective, CNs support the thesis of neural and swarm architectures having similar cognitive abilities.

References

Aron, S., Deneubourg, J. L., Goss, S., and Pasteels, J. M. (1990). Functional self-organization illustrated by inter-nest traffic in ants: The case of the argentinian ant. In *Biological Motion*, W. Alt and G. Hoffman, (Eds.). Lecture Notes in BioMathematics, vol. 89. Springer, Berlin, 533–547. URL http://tinyurl.com/ye28smr.

Balkenius, C., Zlatev, J., Brezeal, C., Dautenhahn, K., and Kozima, H., Eds. (2001). *Proceedings of the First International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*. Vol. 85. Lund University Cognitive Studies, Lund, Sweden. URL http://www.lucs.lu.se/LUCS/085/.

Bar-Yam, Y. (2004). Multiscale variety in complex systems. *Complexity* **9** (4): 37–45. URL http://necsi.org/projects/yaneer/multiscalevariety.pdf.

Blum, C. and Socha, K. (2005). Training feed-forward neural networks with ant colony optimization: An application to pattern classification. *Hybrid Intelligent Systems, International Conference on*: 233–238. URL http://dx.doi.org/10.1109/ICHIS.2005.104.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York.

Chen, Y., Yang, B., and Dong, J. (2004). Evolving flexible neural networks using ant programming and pso algorithm. *Advances in Neural Networks –ISNN 2004*: 211–216. URL http://dx.doi.org/10.1007/b99834.

Cook, M. (2004). Universality in elementary cellular automata. *Complex Systems* **15** (1): 1–40.

Couzin, I. D. (2009). Collective cognition in animal groups. *Trends in Cognitive Sciences* **13** (1): 36 – 43. URL http://dx.doi.org/10.1016/j.tics.2008.10.002.

Dorigo, M. (2007). Ant colony optimization. *Scholarpedia* **2** (3): 1461. URL http://www.scholarpedia.org/article/Ant_colony_optimization.

Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science* **44** (2-3): 243–278. URL http://dx.doi.org/10.1016/j.tcs.2005.05.020.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991). Positive feedback as a search strategy. Tech. Rep. 91-016, Dipartimento di Elettronica, Politecnico di Milano.

Dorigo, M., Montes de Oca, M. A., and Engelbrecht, A. (2008). Particle swarm optimization. *Scholarpedia* **3** (11): 1486. URL http://www.scholarpedia.org/article/Particle_swarm_optimization.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

Fontana, W. (2002). Modelling 'evo-devo' with RNA. *BioEssays* **24** (12): 1164–1177. URL http://tinyurl.com/ykdbdpe.

Gershenson, C. (2002). Classification of random Boolean networks. In *Artificial Life VIII: Proceedings of the Eight International Conference on Artificial Life*, R. K. Standish, M. A. Bedau, and H. A. Abbass, (Eds.). MIT Press, pp. 1–8. URL http://alife8.alife.org/proceedings/sub67.pdf.

Gershenson, C. (2004a). Cognitive paradigms: Which one is the best? *Cognitive Systems Research* **5** (2) (June): 135–156. URL http://dx.doi.org/10.1016/j.cogsys.2003.10.002.

Gershenson, C. (2004b). Updating schemes in random Boolean networks: Do they really matter? In *Artificial Life IX Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R. A. Watson, (Eds.). MIT Press, pp. 238–243. URL http://uk.arxiv.org/abs/nlin.AO/0402006.

Gershenson, C. (2007a). *Design and Control of Self-organizing Systems*. CopIt Arxives, Mexico. http://tinyurl.com/DCSOS2007. URL http://tinyurl.com/DCSOS2007.

Gershenson, C. (2007b). The world as evolving information. In *Proceedings of International Conference on Complex Systems ICCS2007*, Y. Bar-Yam, (Ed.). URL http://uk.arxiv.org/abs/0704.0304.

Gershenson, C. and Heylighen, F. (2003). When can we call a system self-organizing? In *Advances in Artificial Life, 7th European Conference, ECAL 2003 LNAI 2801*, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, (Eds.). Springer, Berlin, pp. 606–614. URL http://uk.arxiv.org/abs/nlin.AO/0303020.

Hölldobler, B. and Wilson, E. O. (1990). *The Ants*. Belknap Press.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences* **79** (8): 2554. URL http://www.pnas.org/content/79/8/2554.abstract.

Hopfield, J. (1988). Artificial neural networks. *Circuits and Devices Magazine, IEEE* **4** (5) (Sep): 3–10. URL http://dx.doi.org/10.1109/101.8118.

Johnson, J. (2010). *Hypernetworks in the Science of Complex Systems*. Series on Complexity Science, vol. 1. World Scientific. URL http://www.worldscibooks.com/chaos/p533.html.

Juárez Martínez, G., McIntosh, H. V., Seck Tuoh Mora, J. C., and Chapa Vergara, S. V. (2007). Rule 110 objects and other collision-based constructions. *Journal of Cellular Automata* **2** (3): 219–242.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*. IEEE Press, Piscataway, NJ, pp. 1942–1948. URL http://tinyurl.com/y8ho5cr.

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

Marshall, J. A., Bogacz, R., Dornhaus, A., Planqué, R., Kovacs, T., and Franks, N. R. (2009). On optimal decision-making in brains and social insect colonies. *Journal of the Royal Society Interface*. URL http://dx.doi.org/10.1098/rsif.2008.0511.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology* **5** (4): 115–133. URL http://dx.doi.org/10.1007/BF02478259.

Mozafari, B., Ranjbar, A. M., Amraee, T., Mirjafari, M., and Shirani, A. R. (2006). A hybrid of particle swarm and ant colony optimization algorithms for reactive power market simulation. *Journal of Intelligent and Fuzzy Systems* **17** (6) (January): 557–574. URL http://tinyurl.com/ygx63rc.

Munteanu, A. and Solé, R. V. (2008). Neutrality and robustness in evo-devo: Emergence of lateral inhibition. *PLoS Comput Biol* **4** (11) (11): e1000226. URL http://dx.doi.org/10.1371%2Fjournal.pcbi.1000226.

Newman, M., Barabási, A.-L., and Watts, D. J., Eds. (2006). *The Structure and Dynamics of Networks*. Princeton Studies in Complexity. Princeton University Press.

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review* **45**: 167–256. URL http://arxiv.org/abs/cond-mat/0303516.

Passino, K. M., Seeley, T. D., and Visscher, P. K. (2008). Swarm cognition in honey bees. *Behavioral Ecology and Sociobiology* **62** (3) (January): 401–414. URL http://dx.doi.org/10.1007/s00265-007-0468-1.

Prokopenko, M., Boschetti, F., and Ryan, A. (2009). An information-theoretic primer on complexity, self-organisation and emergence. *Complexity* **15** (1): 11 – 28. URL http://dx.doi.org/10.1002/cplx.20249.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics* **21** (4): 25–34. URL http://www.red3d.com/cwr/papers/1987/boids.html.

Reznikova, Z. (2007). *Animal Intelligence From Individual to Social Cognition*. Cambridge University Press.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, Eds. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press.

Ryabko, B. and Reznikova, Z. (2009). The use of ideas of information theory for studying "language" and intelligence in ants. *Entropy* **11** (4): 836–853. URL http://dx.doi.org/10.3390/e11040836.

Stahl, G. (2006). *Group Cognition: Computer Support for Building Collaborative Knowledge*. MIT Press.

Trianni, V. and Tuci, E. (2009). Swarm cognition and artificial life. In *Advances in Artificial Life. Proceedings of the 10th European Conference on Artificial Life (ECAL 2009)*.

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2* **42**: 230–265. URL http://www.abelard.org/turpap2/tp2-ie.asp.

von Neumann, J. (1966). *The Theory of Self-Reproducing Automata*. University of Illinois Press. Edited by A. W. Burks.

Wang, Z., Durst, G. L., Eberhart, R. C., Boyd, D. B., and Miled, Z. B. (2004). Particle swarm optimization and neural network application for qsar. In *In HiCOMB*. pp. 26–30.

Wolfram, S. (1986). *Theory and Application of Cellular Automata*. World Scientific.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media. URL http://www.wolframscience.com/thebook.html.

Wolpert, D. and Macready, W. (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* **1** (1): 67–82.

Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search. Tech. Rep. SFI-WP-95-02-010, Santa Fe Institute. URL http://tinyurl.com/yz274ej.

Wuensche, A. (1998). Discrete dynamical networks and their attractor basins. In *Complex Systems '98*, R. Standish, B. Henry, S. Watt, R. Marks, R. Stocker, D. Green, S. Keen, and T. Bossomaier, (Eds.). University of New South Wales, Sydney, Australia, pp. 3–21. URL http://tinyurl.com/y6xh35.

Wuensche, A. and Lesser, M. (1992). *The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*. Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, Reading, MA.