Toward Nature-Inspired Computing

NIC-based systems utilize autonomous entities that self-organize to achieve the goals of systems modeling and problem solving.

By Jiming Liu and K.C. Tsui

Nature-inspired computing (NIC) is an emerging computing paradigm that draws on the principles of self-organization and complex systems. Here, we examine NIC from two perspectives. First, as a way to help explain, model, and characterize the underlying mechanism(s) of complex real-world systems by formulating computing models and testing hypotheses through controlled experimentation. The end product is a potentially deep understanding or at least a better explanation of the working mechanism(s) of the modeled system. And second, as a way to reproduce autonomous (such as lifelike) behavior in solving computing problems. With detailed knowledge of the underlying mechanism(s), simplified abstracted autonomous lifelike behavior can be used as a model in practically any general-purpose problem-solving strategy or technique.

Neither objective is achievable without formulating a model of the factors underlying the system. The modeling process can begin with a theoretical analysis from either a macroscopic or microscopic view of the system. Alternatively, the application developer may adopt a blackbox or whitebox approach. Blackbox approaches (such as Markov models and artificial neural networks) normally do not reveal much about their working mechanism(s). On the other hand, whitebox approaches (such as agents with bounded rationality) are more useful for explaining behavior [7].

The essence of NIC formulation involves conceiving a computing system operated by population(s) of autonomous entities. The rest of the system is referred to as the environment. An autonomous entity consists of a detector (or set of detectors), an effector (or set of effectors), and a repository of local behavior rules (see Figure 1) [5, 8].

A detector receives information related to its neighbors and to the environment. For example, in a simulation of a flock of birds, this information would include the speed and direction the birds are heading and the distance between the birds in question. The

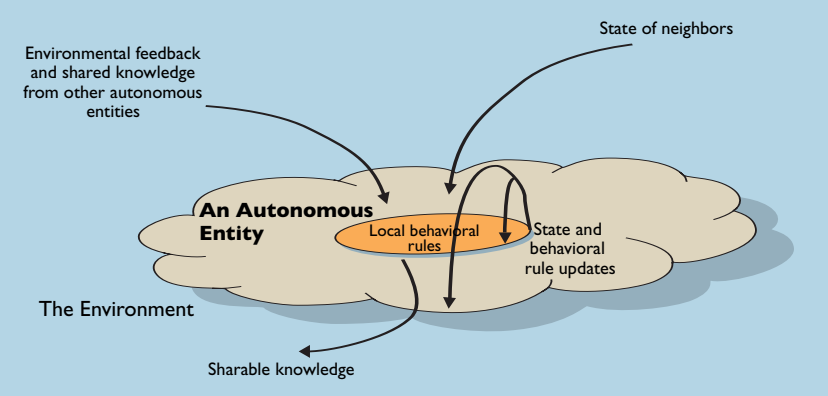

Figure 1. Modeling an autonomous entity in a NIC-based system.

details of the content and format of the information must be defined according to the system to be modeled or to the problem to be solved. The notion of neighbors may be defined in terms of position (such as the bird(s) ahead, to the left, and to the right), distance (such as a radial distance of two grids), or both (such as the birds up to two grids ahead of the nominal viewpoint bird).

Environmental information conveys the status of a certain feature of interest to an autonomous entity. The environment can also help carry sharable local knowledge. The effector of an autonomous entity refers collectively to the device for expressing actions. Actions can be changes to an internal state, an external display of certain behaviors, or changes to the environment the entity inhabits. An important role of the effector, as part of the local behavior model, is to facilitate implicit information sharing among autonomous entities.

Central to an autonomous entity are the rules of behavior governing how it must act or react to the information collected by the detector from the environment and its neighbors. These rules determine into what state the entity should change and also what local knowledge should be released via the effector to the environment. An example of sharable local knowledge is the role pheromones play in an ant colony. It is untargeted, and the communication via the environment is undirected; any ant can pick up the information and react according to its own behavior model.

In order to adapt itself to a problem without being explicitly told what to do in advance, an autonomous entity must modify the rules of its behavior over time. This ability, responding to local changing conditions, is known as the individual's learning capability. Worth noting is that randomness plays a part in the decision-making process of an autonomous entity despite the presence of a rule set. It allows an autonomous entity to explore uncharted territory despite evidence that it should exploit only a certain path. On the other hand, randomness helps the entity resolve conflict in the presence of equal support for suggestions to act in different ways in its own best interests and avoid being stuck by randomly choosing an action in local optima.

The environment acts as the domain in which autonomous entities are free to roam. This is a static view of the environment. The environment of a NIC system can also act as the "noticeboard" where the autonomous entities post and read local information. In this dynamic view, the environment is constantly changing For example, in the N-queen constraint satisfaction problem [7], the environment can tell a particular queen on a chessboard how many constraints are violated in her neighborhood after a move is made. In effect, this violations, or conflicts, report translates a global goal into a local goal for a particular entity. The environment also keeps the central clock that helps synchronize the actions of all autonomous entities, as needed.

Before exploring examples of NIC for characterizing complex behavior or for solving computing problems, we first highlight the central NIC ideas, along with common NIC characteristics, including autonomous, distributed, emergent, adaptive, and self-organized, or ADEAS [5]:

Autonomous. In NIC systems, entities are individuals with bounded rationality that act independently. There is no central controller for directing and coordinating individual entities. Formal computing models and techniques are often used to describe how the entities acquire and improve their reactive behavior, based on their local and/or shared utilities, and how the behavior and utilities of the entities become goal-directed.

Distributed. Autonomous entities with localized decision-making capabilities are distributed in a heterogeneous computing environment, locally interacting among themselves to exchange their state information or affect the states of others. In distributed problem solving (such as scheduling and optimization), they continuously measure, update, and share information with other entities following certain predefined protocols.

Emergent. Distributed autonomous entities collectively exhibit complex (purposeful) behavior not present or predefined in the behavior of the autonomous entities within the system. One interesting issue in studying the emergent behaviors that leads to some

desired computing solutions (such as optimal resource allocation) is how to mathematically model and measure the interrelationships among the local goals of the entities and the desired global goal(s) of the NIC system in a particular application.

Adaptive. Entities often change their behavior in response to changes in the environment in which they are situated. In doing so, they utilize behavioral adaptation mechanisms to continuously evaluate and fine-tune their behavioral attributes with reference to their goals, as well as to ongoing feedback (such as intermediate rewards). Evolutionary approaches may be used to reproduce high-performing entities and eliminate poor-performing ones.

Self-organized. The basic elements of NIC-based systems are autonomous entities and their environment. Local interactions among them are the most powerful force in their evolution toward certain desired states. Self-organization is the essential process of a NIC system's working mechanism. Through local interactions, these systems self-aggregate and amplify the outcome of entity behavior.

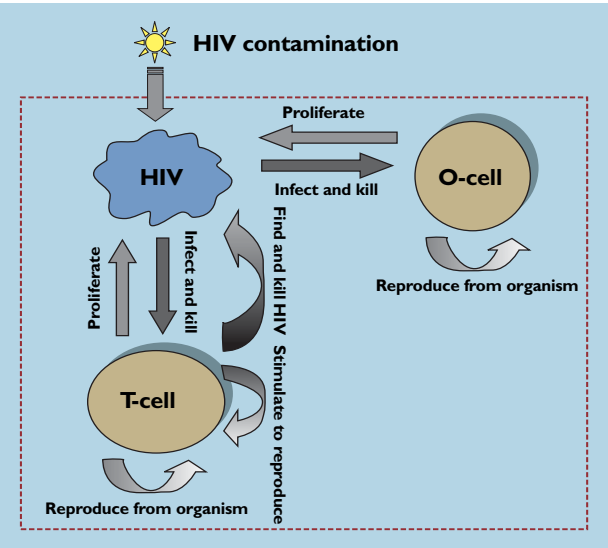

Figure 2. Modeling HIV, T-cells, and O(ther) cells in a NIC-based system.

Characterizing Complex Behavior

A complex system can be analyzed and understood in many different ways. The most obvious is to look at it from the outside, observing its behaviors and using models to try to identify and list them. Assumptions about unknown mechanisms must be made to start the process. Given observable behaviors of the desired system, NIC designers verify the model by comparing its behavior with the desired features. This process is repeated several times before a good, though not perfect, prototype can be found. Apart from obtaining a working model of the desired system, an important by-product is the discovery of the mechanisms that were unknown when the design process began.

The human immune system is an example of a highly sensitive, adaptive, self-regulated complex system involving numerous interactions among a vast number of cells of different types. Despite numerous clinical case studies and empirical findings [1], the working mechanism underlying the complex process of, say, HIV invasion and the erosion and eventual crash of the immune system (including how the local interactions in HIV, T-cells, and B-cells affect the process) are still not fully understood (characterized and predicted).

The usefulness of conventional modeling and simulation technologies is limited due to computational scale and costs. Understanding and modeling complex systems (such as the human immune system) is a major challenge for the field of computing for two main reasons: the task of computing is seamlessly carried out in a variety of physical embodiments, and no single multipurpose or dedicated machine is able to accomplish the job. The key to success for simulating self-regulated complex systems lies in the large-scale deployment of computational entities or agents able to autonomously make local decisions and achieve collective goals.

Seeking to understand the dynamics of the immune system during an HIV attack, NIC researchers can use a 2D lattice to build a NIC model. The lattice is circular so the edges wrap around one another. Each site can be inhabited by HIV, as well as by immune cells. HIV and immune cells behave in four main ways:

Interaction. T-cells recognize HIV by its signature (protein structure); HIV infects and kills cells;
Proliferation. Reactions stimulate lymphoid tissue to produce more T-cells, which are reproduced naturally;
Death. Besides being killed by drugs and other deliberate medical intervention, HIV and T-cells die naturally; and
Diffusion. HIV diffuses from densely populated sites to neighboring sites (see Figure 2).

Figure 3 outlines the temporal emergence of three-stage dynamics in HIV infection generated from the NIC model [12]:

Before B. Primary response;
B~C. Clinical latency; and
After D. Onset of AIDS.

At A, the HIV population reaches a maximum point. Starting from C, the mechanism that decreases the natural ability of an organism to reproduce T-cells is triggered. These NIC-generated results are consistent with empirically observed phenomena [1]. Experiments in [12] have also found that AIDS cannot break out if HIV destroys only T-cells without weakening the T-cell reproduction mechanism. The emergence in "shape space" indicates it is because of HIV's fast mutation that the immune system cannot eradicate HIV as readily as it does other invaders. These discoveries are helping immunology researchers understand the dynamics of HIV-immune interaction.

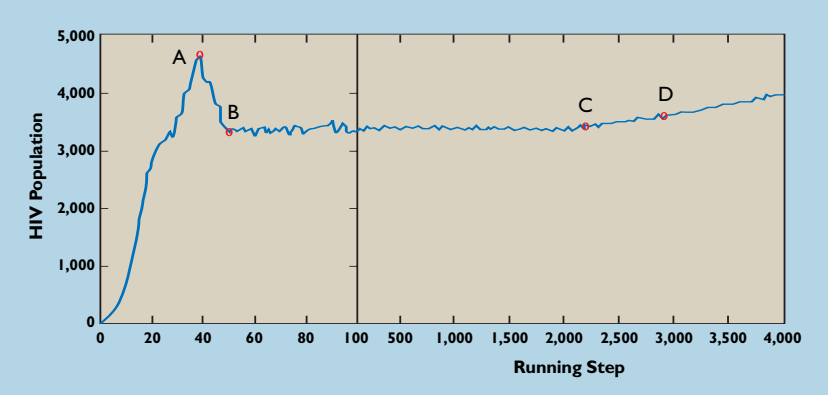

Figure 3. Simulation results on an HIV population during several phases of AIDS development.

The NIC approach to systems modeling starts from a microscopic view of the immune system. The elements of the model are the basic units—HIV and immune cells—of the immune system. The model aims to capture the essence of the immune system, though simplification is inevitable. Note that the autonomous entities in the model that belong to the same species types normally have a similar set of behavior rules. The only difference among them is the parameters of the rules, which may be adapted throughout the lifetime of the entities. Probabilistic selection of certain behavior is also common in the entities. It must be emphasized that the environment of the model can also be viewed as a unique entity in the model, with its own behavior rules.

Self-Organized Web Regularities

Researchers have identified several self-organized regularities related to the Web, ranging from its growth and evolution to usage patterns in Web surfing. Many such regularities are best represented by characteristic distributions following a Zipf law or a power law. Random-walk models [4] have been used to simulate some statistical regularities empirically observed on the Web. However, these models do not relate the emergent regularities to the dynamic interactions between users and the Web, nor do they reflect the interrelationships between user behavior and the contents or structure of the Web. User interest and motivation in navigating the Web are among the most important factors determining user navigation behavior.

As part of the NIC approach to regularity characterization, [6] proposed a computational model of Web surfing that includes user characteristics (such as interest profiles, motivations, and navigation strategies). Users are viewed as information-foraging entities inhabiting the Web environment or as a collection of Web sites connected by hyperlinks. When an entity finds certain Web sites with content related to its topic(s) of interest, it will be motivated to search sites deeper into the Web. On the other hand, when an entity finds no interesting information after a certain amount of foraging or finds enough content to satisfy its interest, it stops foraging and goes offline, leaving the Web environment.

Experiments in [6] classified users into three groups: recurrent users familiar with the Web structure; rational users new to a particular Web site but who know what they are looking for; and random users with no strong intention to retrieve information but are just "wandering around." The results, which used both synthetic and empirical data from visitors to NASA Web site(s), showed that the foraging agent-entity-oriented model generates power-law distributions in surfing and link-click-frequency, similar to those found in the real world and hence offer a whitebox explanation of self-organized Web regularities.

Solving Computing Problems

The key factors contributing to the success of these NIC-based models are the distinctive characteristics of their elements. Marvin Minsky of MIT suggested in his 1986 book *Society of Mind* that "To explain the mind, we have to show how minds are built from mindless stuff, from parts that are much smaller and simpler than anything we'd consider smart." So, if we want to formulate a problem-solving strategy based on some observation from nature, how and where should we begin? To formulate a NIC problem-solving system, we must identify and gain a deep understanding of a working system in the natural or physical world from which models can be extracted. As with complex-systems modeling, the

abstracted behavior of the working system becomes the property of the elements to be modeled.

Basing their approach on the general principles of survival of the fittest (whereby poor performers are eliminated) and the "law of the jungle" (whereby weak performers are eaten by stronger ones), several NIC systems have been devised [7, 9] to solve some well-known constraint-satisfaction problems. One is the N-queen problem, in which N queens are placed on an N x N chessboard, so no two queens ever appear in the same row, column, or diagonal. Based on the rules of the problem, a NIC model is formulated in the following way: Each queen is modeled as an autonomous entity in the system, and multiple queens are assigned to each row of the chessboard (a grid environment). This process allows for competition among the queens in the same row, so the queen with the best strategy survives. The system calculates the number of violated constraints for each position on the grid. This represents the environmental information all queens can access when making decisions about where to move, with possible movements being restricted to positions in the same row.

Three movement strategies are possible: random-move (involving the random selection of a new position for a queen); least-move (involving selection of the position with the least number of violations, or conflicts); and coop-move (promoting cooperation among the queens by eliminating certain positions in which one queen's position may create conflicts with other queens). All three are selected probabilistically.

This NIC system gives an initial amount of energy to each queen. Like a character in a video game, a queen "dies" if its energy falls below a predefined threshold. A queen's energy level changes in one of two ways: losing it to the environment and absorbing it from another queen. When a queen moves to a new position that violates the set constraint with *m* queens, it loses *m* units of energy. This also causes the queens that attack this new position to lose one unit of energy. The intention is to encourage the queens to find a position with the fewest violations, or conflicts. The law of the jungle is implemented by having two or more queens occupy the same grid position and fight over it. The queen with the greatest amount of energy wins and eats the loser(s) by absorbing all its (their) energy. This model efficiently solves the N-queen problem with only a moderate amount of computation.

In the commonly used version of a genetic algorithm [3], a member of the family of evolutionary algorithms, the process of sexual evolution is simplified to selection, recombination, and mutation, without the explicit identification of male and female

As with complex-systems modeling, the abstracted behavior of the working system becomes the property of the elements to be modeled.

(such as in the gene pool). John Holland of the University of Michigan, in his quest to develop a model to help explain evolution, has developed a genetic algorithm for optimization. The basic unit in this artificial evolution is a candidate solution to the optimization problem, commonly termed a chromosome. A genetic algorithm has a pool of them. Interactions among candidate solutions are achieved through artificial reproduction where operations mimicking natural evolution allow the candidate solutions to produce offspring that carry part of either parent (crossover) with occasional variation (mutation). While reproduction can be viewed as the cooperative side of all the chromosomes, competition among chromosomes for a position in the next generation directly reflects the principle of survival of the fittest.

On the other hand, evolutionary autonomous agents [10] and evolution strategies [11] are closer to asexual reproduction, with the addition of constraints on mutation and the introduction of mutation operator evolution, respectively. Despite this simplification and modification, evolutionary algorithms capture the essence of natural evolution and are proven global multi-objective optimization techniques. Another successful NIC algorithm that has been applied in similar domains is the Ant System [2], which mimics the food-foraging behavior of ants.

Autonomy-Oriented Computing

As a concrete manifestation of the NIC paradigm, autonomy-oriented computing (AOC) has emerged as a new field of computer science to systematically explore the metaphors and models of autonomy offered in nature (such as physical, biological, and social entities of varying complexity), as well as their

role in addressing practical computing needs. It studies emergent autonomy as the core behavior of a computing system, drawing on such principles as multi-entity formulation, local interaction, nonlinear aggregation, adaptation, and self-organization [5, 8].

Three general approaches help researchers develop AOC systems: AOC-by-fabrication, AOC-by-prototyping, and AOC-by-self-discovery. Each has been found to be promising in several application areas [6, 7, 10, 12]. Work on AOC in our research laboratory over the past decade [5, 8] has opened up new ways to understand and develop NIC theories and methodologies. They have provided working examples that demonstrate the power and features of the NIC paradigm toward two main goals: characterizing emergent behavior in natural and artificial systems involving a large number of self-organizing, locally interacting entities; and solving problems in large-scale computation, distributed constraint satisfaction, and decentralized optimization [5].

Conclusion

The NIC paradigm differs from traditional imperative, logical, constraint, object-oriented, and component-based paradigms, not only in the characteristics of its fundamental concepts and constructs, but in the effectiveness and efficiency of the computing that can be achieved through its ADEAS characteristics. NIC approaches have been found most effective in dealing with computational problems characterized by the following dimensions:

High complexity. Problems of high complexity (such as when the system to be characterized involves a large number of autonomous entities or the computational computing problem to be solved involves large-scale, high-dimension, highly nonlinear interactions/relationships, and highly interrelated/constrained variables);

Locally interacting problems. They are not centralized or ready or efficient enough for batch processing;

Changing environment. The environment in which problems are situated is dynamically updated or changes in real time; and

Deep patterns. The goal of modeling and analysis is not to extract some superficial patterns/relationships, data transformation, or association from one form to another, but to discover and understand the deep patterns (such as the underlying mechanisms and processes that produce the data in the first place or help explain their cause and origin).

We will continue to see new NIC theories and methodologies developed and learn to appreciate their wide-ranging effect on computer science, as well as on other disciplines, including sociology, economics, and the natural sciences. Promising applications will help explain gene regulatory networks and drug-resistance mechanisms for anti-cancer drug design, predict the socioeconomic sustainability of self-organizing online markets or communities, and perform real-time autonomous data processing in massive mobile sensor networks for eco-geological observations. C

References

1. Coffin, J. HIV population dynamics in vivo: Implications for genetic variation, pathogenesis, and therapy. *Science 267* (1995), 483–489.
2. Dorigo, M., Maniezzo, V., and Colorni, A. The Ant System: Optimization by a colony of cooperative agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B, 26,* 1 (1996), 1–13.
3. Holland, J. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, MA, 1992.
4. Huberman, B., Pirolli, P., Pitkow, J., and Lukose, R. Strong regularities in World Wide Web surfing. *Science 280* (Apr. 3, 1997), 96–97.
5. Liu, J., Jin, X., and Tsui, K. *Autonomy-Oriented Computing: From Problem Solving to Complex Systems Modeling.* Kluwer Academic Publishers/Springer, Boston, 2005.
6. Liu, J., Zhang, S., and Yang, J. Characterizing Web usage regularities with information foraging agents. *IEEE Transactions on Knowledge and Data Engineering 16,* 5 (2004), 566–584.
7. Liu, J., Han, J., and Tang, Y. Multi-agent-oriented constraint satisfaction. *Artificial Intelligence 136,* 1 (2002), 101–144.
8. Liu, J. *Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization and Adaptive Computation.* World Scientific Publishing, Singapore, 2001.
9. Liu, J. and Han, J. ALife: A multi-agent computing paradigm for constraint satisfaction problems. *International Journal of Pattern Recognition and Artificial Intelligence 15,* 3 (2001), 475–491.
10. Liu, J., Tang, Y., and Cao, Y. An evolutionary autonomous agents approach to image feature extraction. *IEEE Transactions on Evolutionary Computation 1,* 2 (1997), 141–158.
11. Schwefel, H.-P. *Numerical Optimization of Computer Models.* John Wiley & Sons, Inc., New York, 1981.
12. Zhang, S. and Liu, J. A massively multi-agent system for discovering HIV-immune interaction dynamics. In *Proceedings of the First International Workshop on Massively Multi-Agent Systems* (Kyoto, Japan, Dec. 10–11). Springer, Berlin, 2004.

Jiming Liu (jiming@uwindsor.ca) is a professor in and director of the School of Computer Science at the University of Windsor, Windsor, Ontario, Canada.

K.C. Tsui (tsuikc@comp.hkbu.edu.hk) is an IT manager in the Technical Services and Support Department of Hongkong and Shanghai Banking Corporation, Hong Kong, China.

This work is supported by the Research Grants Council of Hong Kong (RGC/HKBU2121/03E and RGC/HKBU2040/02E) and the Natural Sciences and Engineering Research Council of Canada, as carried out at Jiming Liu's Autonomy-Oriented Computing/Autonomous Agents and Multi-Agent Systems research laboratory.